
Universidade de Aveiro Departamento de Electrónica e Telecomunicações
2003

Salviano Filipe Silva Pinto Soares

Algoritmos de Reconstrução de Sinal e Correcção de Erros

Universidade de Aveiro Departamento de Electrónica e Telecomunicações
2003

Salviano Filipe Silva Pinto Soares

Algoritmos de Reconstrução de Sinal e Correcção de Erros

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Electrotécnica, realizada sob a orientação científica do Dr. Paulo Jorge dos Santos Gonçalves Ferreira, Professor Catedrático do Departamento de Electrónica e Telecomunicações da Universidade de Aveiro

O júri

Presidente

Prof. Dr. José Joaquim Cristino Teixeira Dias
Professor Catedrático da Universidade de Aveiro por delegação da Reitora da Universidade de Aveiro

Prof. Dr. Luís António Serralva Vieira de Sá
Professor Catedrático da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Prof. Dr. Paulo Jorge dos Santos Gonçalves Ferreira
Professor Catedrático da Universidade de Aveiro

Prof. Dr. Aníbal João de Sousa Ferreira
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

Prof. Dr. José Manuel Bioucas Dias
Professor Auxiliar do Instituto Superior Técnico da Universidade Técnica de Lisboa

Prof. Dr. José Manuel Neto Vieira
Professor Auxiliar da Universidade de Aveiro

Agradecimentos

Começo por expressar os meus sinceros agradecimentos ao meu orientador, Prof. Paulo Jorge Ferreira, pela disponibilidade sempre demonstrada. Foi para mim um enorme prazer os encontros que mantivemos ao longo destes últimos anos. O meu obrigado pelos seus ensinamentos.

Aos meus caros colegas, Francisco Pereira, João Barroso e Manuel Cabral, quero expressar o meu apreço pelo bom ambiente desde sempre.

À Universidade de Trás-os-Montes e Alto Douro, na pessoa do seu Magnífico Reitor, agradeço as facilidades que sempre me foram concedidas.

Finalmente, à minha esposa e minha filha, meus pais e meu irmão, uma palavra especial pelo apoio e presença reconfortante ao longo deste trabalho.

Resumo

Este trabalho situa-se na intersecção de três áreas: os códigos de controlo de erros, os métodos numéricos e o processamento de sinal (em especial no que diz respeito às técnicas de reconstrução de sinal e interpolação). Procura-se explorar as relações entre estes temas, desenvolvendo o potencial da interpolação e reconstrução de sinal no contexto da descodificação de códigos reais.

Os códigos reais diferem dos tradicionais, formulados em termos de corpos finitos, pelo facto de se basearem no corpo real ou no corpo complexo, e por recorrerem a técnicas clássicas de processamento digital de sinal, como a FFT. Têm vantagens sobre os códigos tradicionais no que diz respeito à utilização de aritmética real, à possibilidade de utilização de processadores de sinal, à inexistência de restrições quanto aos tamanhos de bloco, e à tolerância a ruído de fundo. A desvantagem principal é a existência de erros de arredondamento e as dificuldades numéricas decorrentes.

Neste trabalho examinam-se as alternativas de descodificação de códigos reais. Exploram-se duas formulações do problema, uma no domínio do tempo e outra no domínio da frequência, que conduzem a matrizes de tamanho mínimo para cada problema. Às equações obtidas foram aplicados de forma sistemática um conjunto de métodos directos, iterativos e semi-iterativos. Os resultados permitiram identificar a estratégia óptima de descodificação de códigos reais, que se concluiu depender criticamente dos detalhes do problema. A forma como a natureza do padrão de erros, o número total de erros, a redundância do código e o tamanho do bloco determinam a natureza do descodificador óptimo é estudada.

Abstract

The subject of this work belongs to the intersection of three areas: error control coding, numerical methods and signal processing (particularly signal reconstruction and interpolation). The work seeks to explore the connections among these areas, in reference to the decoding of real number codes.

Real number codes differ from the usual error control codes mainly in what concerns the underlying algebraic fields: real number codes depend on the real or complex fields, whereas traditional codes rely on finite (or Galois) fields. The advantages of real number codes include the convenience of using real arithmetic, the possibility of DSP-based implementation, the lack of restrictions regarding block size, and the tolerance to low-level noise. The main disadvantage is the existence of round off error, and the consequent numerical difficulties.

This work examines several alternative ways of decoding real number codes. It explores two approaches to the problem, one in the time domain, the other in the frequency domain, both leading to minimum dimension equations. To the equations that follow from each of these formulations a number of iterative, noniterative and semi-iterative algorithms are systematically applied. The results reveal crucial information concerning the optimum decoding strategy for real number codes, and how the error pattern, the total number of errors, the code rate, and the block size determine the nature of the optimal decoding algorithm.

À Manuela
e à nossa doce filha
Carolina

…, pois, e a invenção da descoberta da 2# *blue note* 3 `!...

…*Salblues*…

Índice

Índice de figuras

Índice de tabelas

CAPÍTULO 1

INTRODUÇÃO

1.1 Enquadramento e motivação

A disponibilidade, desempenho e custo do micro-computador digital tornou inevitável a conversão dos antigos sistemas analógicos em digitais. O desenvolvimento natural do processamento digital dos dados teve como factor externo aliado a pressão económica que funcionou como catalizador para a sua rápida evolução. Estes factores contribuíram de um modo significativo para tornar a "filosofia digital" dominante nos dias de hoje.

A representação digital de sinais possibilita a correcção de erros, mesmo quando se trabalha com aritmética real ou complexa [Rudin87]. Por exemplo, um sinal de banda limitada pertencente ao espaço das funções de quadrado integrável ou de energia finita L_2, pode ser reconstruído a partir das suas amostras se estas forem tomadas a uma frequência de amostragem maior do que a frequência de *Nyquist*, mesmo desconhecendo um número finito dessas amostras [Marks93, Zayed93, Gröchenig93A, Ferreira92, Strohmer95, Benedetto00].

Quando as amostras são de um sinal aproximadamente limitado em frequência, com o auxílio de uma transformada linear podem ser convertidos em dados limitados em frequência com respeito a essa transformada linear. Este processo introduz ou aproveita a redundância nos dados, e possibilita assim a detecção e correcção de eventuais erros. Estas técnicas são exemplos de códigos reais, isto é, técnicas para controlar (detectar e eventualmente corrigir) erros em sinais digitais, baseadas em aritmética real ou

complexa. O problema pode ser formulado neste contexto e conduz-nos a um conjunto de equações lineares [Ferreira94D].

Garantir teoricamente a existência e unicidade da solução das equações lineares pode não ser suficiente na prática. Considere-se por exemplo o sistema de equações lineares genérico na sua forma matricial $Ax=b$. Teoricamente, existe uma solução única quando a matriz A é não-singular [Horn85]. Contudo, a ordem do sistema, ou a existência de um valor próprio da matriz próximo de zero, podem tornar impraticável obter a sua solução. Noutro sentido, mas atendendo aos mesmos pressupostos, taxas de convergência baixas em métodos iterativos ou a necessidade de um sistema informático com poder computacional elevado podem constituir outras condicionantes para que a resolução do problema em tempo útil não seja possível. Estas considerações são cruciais no contexto de códigos reais.

Considere-se o problema de transmitir informação de forma a ser possível descodificá-la no receptor com um probabilidade de erro muito pequena. A Teoria dos Códigos de Correcção de Erros (TCCE) aborda este assunto. A utilização de códigos de correcção de erros reduz-se à adição cuidadosa de redundância aos dados antes da modulação de modo a que no receptor se detectem ou eventualmente se corrijam erros introduzidos pelo processo de transmissão. Na prática, pretendem encontrar-se sequências de símbolos de entrada cuja distância entre si seja suficiente para que na saída as suas versões corrompidas possam ser distinguidas. Assim, na mensagem inicial todos os símbolos são possíveis, enquanto que no receptor apenas um subconjunto pode ocorrer. É possível para um conjunto de símbolos calcular a probabilidade com que no receptor se cometem erros. Shannon demonstrou que quando se transmite com taxas inferiores à capacidade do canal se pode obter uma probabilidade de erro arbitrariamente baixa [Shannon48, Shannon49]. O código ideal corrigiria todos os padrões de erros e com performances perto de limite de Shannon. A grande parte dos códigos de correcção de erros usa corpos finitos normalmente designados $GF(2^p)$ [Brison97, Rotman98]. Se por um lado a álgebra associada se torna mais complexa, ou pelo menos não tão usual, por outro os erros de representação e de arredondamentos são nulos.

Ao contrário dos corpos finitos, a aritmética no corpo real não se pode realizar de forma exacta. Uma precisão finita imposta à partida pode afectar de forma crítica a estabilidade numérica de qualquer algoritmo de reconstrução e condicionar o seu

desempenho. Por outro lado em códigos reais não existem limitações ao tamanho do bloco, nem necessidades de hardware dedicado ou software específico, ao contrário dos códigos implementados em corpos finitos.

A localização dos erros pode ditar diferentes abordagens ao seu tratamento. Por exemplo, admita-se um sistema em que a informação é transmitida bloco a bloco em "pacotes", com um cabeçalho que identifica o destinatário. O facto dos erros ocorrerem no cabeçalho ou no conteúdo, origina metodologias distintas de tratamento de erros já que, enquanto que no primeiro caso os pacotes se desviam do destino, já no segundo a mensagem original chega ao receptor, embora alterada.

Os erros do tipo rajada ou contíguos são muito comuns em transmissões multimédia sobre canais susceptíveis a erros. Este tipo de erros não são fáceis de tratar e podem ser originados por:

- perda de pacotes em redes de comutação de pacotes;
- atraso excessivo em aplicações de tempo real;
- falhas em sistemas de armazenamento (CD-ROM, DVD, suporte magnético).

A retransmissão de pacotes é um mecanismo simples mas nem sempre eficaz. O entrelaçamento dos dados apesar de aumentar o atraso, pode transformar a perda de informação contígua em dispersa, logo mais fácil de corrigir [Yu96, Marvasti97, Cao99]. Utilizar técnicas de controlo de erros baseadas em códigos convencionais [Blahut83] ou turbo códigos [Divsalar95, Rothweiller99] é outra possibilidade para tratar este tipo de problemas.

Idealmente o desempenho dum determinado método genérico não deveria depender da arquitectura onde está a ser executada. Na prática, a diversidade de sistemas disponíveis e linguagens de programação lançaram um novo desafio no sentido de reinventar métodos antigos, ou mesmo apresentar novas criações em função da evolução tecnológica. A existência de computadores multiprocessador capazes de realizar tarefas em simultâneo [Steen94], e a possibilidade de incorporar no projecto de um algoritmo o conceito de paralelismo repartindo assim blocos de código por diferentes processadores [Barret94], reiteram a afirmação anterior. Ainda no mesmo contexto, a possibilidade que alguns compiladores oferecem de seleccionar o tipo de arredondamento ou de optar por precisão simples ou dupla em algumas situações, permite uma adaptação do código ao hardware no sentido de aumentar por exemplo o número de MFLOPS [Frigo99].

Apesar de existirem resultados com concatenação de códigos reais em dois canais interligados por um permutador [Ferreira03], pareceu-nos que a análise de códigos reais com um único canal deveria ser complementada, aprofundando o estudo e avaliações dos métodos de descodificação ou interpolação, incluindo os métodos de dimensão mínima quer no tempo quer na frequência. A tentativa de optimizar e escolher para problemas específicos soluções específicas e o cálculo dos pontos críticos (*break even points*) que determinam os melhores métodos, pode servir de base para o trabalho sobre sistemas baseados em dois canais.

O projecto de técnicas de correcção de erros em transmissão de conteúdos multimédia de alto débito sobre canais heterogéneos sujeitos a erros, dada a ubiquidade das telecomunicações, tornou o assunto actual para a comunidade científica da área. Como exemplos, temos os canais associados a comunicações celulares e redes locais LAN ou PAN sem fio.

Uma possibilidade de tratar este tipo de problemas, nomeadamente em canais limitados em banda, consiste em utilizar técnicas conjuntas de canal/fonte (*joint source-channel coding*) [Azami96, Frias97] que melhoram o desempenho das transmissões ao integrar no mesmo projecto do código a taxa de compressão de bits e capacidade de recuperação de erros necessárias de modo a tornar a comunicação mais robusta.

Na abordagem clássica, no contexto dos corpos de Galois, a redundância necessária para a correcção de erros tem de ser acrescentada aos dados. Um conjunto de novas técnicas de codificação canal/fonte baseadas em estruturas oblíquas (*frames*) reais ou complexas e bancos de filtros sobre-amostrados adicionam redundância aos dados num domínio que é compatível com os próprios dados. Este assunto constituiu uma sessão especial da EUSIPCO-2004 em Setembro com o título "*Robust Transmission of Multimedia Contents*". O trabalho presente insere-se no contexto deste programa tão actual.

1.2 Objectivos

A resolução de sistemas de equações lineares foi desde sempre um problema importante em engenharia. A possibilidade de obter soluções numéricas de um modo eficiente e cada vez mais rápido, fez com que inúmeras aplicações surgissem para os algoritmos que resolvem sistemas de equações lineares.

As restrições de tempo real destes algoritmos sugerem a sua reorganização no sentido de utilizar os recursos de uma forma mais eficiente: os registos, a *cache* ou a memória de um computador contam-se entre alguns deste recursos. Se existir também a possibilidade de adaptação ao hardware no sentido de melhorar o desempenho geral do método, a sua performance será óptima para cada máquina e mais independente das plataformas disponíveis.

Pretende-se para uma classe de problemas de reconstrução de sinais e controlo de erros no corpo real, encontrar um conjunto de métodos que no tempo ou na frequência tenham desempenho óptimo. Note-se que estes problemas incluem a solução de sistemas de equações lineares como um dos passos. Nos trabalhos [Ferreira94B, Ferreira96] demonstra-se a existência da solução dessas equações sob certas condições.

Caso não existam métodos universalmente óptimos, pretende-se identificar as classes de problemas para as quais uma família de métodos é particularmente apropriada e determinar os pontos críticos. No nosso caso, estes são valores limite a partir dos quais se opta por classes de métodos no tempo em detrimento dos na frequência ou vice-versa.

1.3 Resultados originais

Este trabalho contém a primeira investigação sistemática acerca de métodos para a descodificação no tempo e na frequência de códigos reais (e consequentemente para a interpolação de sinais limitados em frequência).

Distinguem-se duas classes de métodos: os primeiros baseiam-se na solução de um sistema de equações de dimensão igual ao número de amostras desconhecidas do sinal no domínio do tempo, que designaremos por n, enquanto que os segundos partem de um sistema de equações de dimensão igual ao número de harmónicos não nulos do sinal, que designaremos por $2M+1$. Como o problema original tanto se pode resolver mediante a solução do problema de ordem n como do problema de ordem $2M+1$, temos à disposição duas formulações equivalentes do problema inicial. Ao número

$$d := \min(n, 2M+1) ,$$

que determina a dimensão do menor sistema de equações para a solução do problema original chamaremos dimensão mínima.

Propôs-se uma nova classificação para os diversos métodos apresentados, tendo em conta a dimensão dos espaços associados e o facto de terem como domínio o tempo ou

a frequência. Esta perspectiva clarificou algumas ligações entre os diferentes algoritmos e permitiu completar o conjunto de métodos com todas as formulações no tempo e na frequência.

O trabalho realça a possibilidade de nos métodos de factorização no tempo ou na frequência, realizar esta apenas uma vez, quando se pretendem resolver múltiplos problemas de estrutura semelhante (como é o caso dos problemas de extrapolação de imagem).

O trabalho permite ainda entender de forma mais completa o modo como se pode explorar a estrutura de Toeplitz. Na frequência isso é sempre possível, se os sinais forem do tipo passa baixo, mas as equações requerem aritmética complexa. No tempo a aritmética é sempre real, se os sinais forem reais, mas a estrutura Toeplitz só existe para determinadas distribuições de amostras desconhecidas. Em particular, existe sempre no caso de problemas de extrapolação.

O estudo sistemático das duas formulações possíveis e a aplicação do conjunto de métodos iterativos, directos e semi-iterativos a cada uma, conduziu a abordagens que, tanto quanto sabemos, são estudadas neste contexto pela primeira vez. No domínio do tempo são exemplos os métodos, Gradientes Conjugados com aceleração Toeplitz (CGT) e Gradientes Conjugados com pré-condicionamento e aceleração Toeplitz (PCGT). Na frequência, temos Cholesky (CHF), Levinson (LEVF), Iteração Simples (SIF) e *Sucessive Overrelaxation* (SORF).

Uma das principais conclusões do trabalho é que o "método ideal" não existe, uma vez que para diferentes combinações dos parâmetros se obtêm diferentes algoritmos "óptimos". Os parâmetros são n, $2M+1$ (englobados no conceito de dimensão mínima) mas também o tamanho do bloco N, a distribuição das amostras desconhecidas e uma multiplicidade de outros factores de muito menor impacto (tamanho da *cache* e outras características arquitecturais por exemplo).

1.4 Organização do trabalho

Este capítulo começou por fazer o enquadramento do tema do trabalho. Em seguida foram apresentados os objectivos e enunciados os resultados que pensamos serem originais.

O capítulo dois, divide-se em duas partes, e é constituído por definições e assuntos quer da Álgebra Linear quer da Análise Numérica usados ao longo de todo o trabalho.

No capítulo três é feita uma revisão de métodos iterativos, semi-iterativos, directos e com pré-condicionamento para a resolução de equações lineares.

Apesar de muitos resultados dos capítulos dois e três serem conhecidos, foram incorporados nesta dissertação para tornarem o texto mais acessível a não especialistas interessados em estudar o assunto.

No capítulo quatro é dada uma perspectiva histórica acerca da TCCE e Teoria da Informação, e faz-se uma analogia entre os primeiros códigos de correcção que eram sobre corpos finitos e os códigos reais/complexos.

O capítulo cinco mostra que é possível resolver os problemas de interpolação no domínio do tempo ou da frequência com métodos denominados de dimensão mínima.

No capítulo seis são apresentados os resultados dos vários algoritmos associados aos códigos DFT para um conjunto de experiências.

No final, são apresentadas as conclusões, mencionando alguns problemas em aberto e apontadas algumas direcções para trabalho futuro.

CAPÍTULO 2

Preliminares

O presente capítulo tem como principais objectivos introduzir a notação e fazer a revisão de conceitos e ferramentas fundamentais que irão ser úteis ao longo do trabalho.

Designaremos o conjunto dos números complexos, reais e inteiros pelas letras $\mathbb{C}$, $\mathbb{R}$ e $\mathbb{Z}$ respectivamente. Sendo assim, $\mathbb{C}^N$, $\mathbb{R}^N$ e $\mathbb{Z}^N$ dizem respeito ao espaço vectorial complexo, real e inteiro de dimensão N. A um elemento destes espaços, genericamente x, chamaremos vector.

Usaremos parêntesis curvo para variáveis (funções) contínuas e parêntesis rectos para variáveis (funções) discretas. Tipicamente, as sequências indexadas ao longo do texto são uma versão amostrada de funções de variável contínua que genericamente podem ser representadas por

$$x[n] = f(nT), \quad n \in \mathbb{Z}, T \in \mathbb{R}.$$

Regra geral, os sinais discretos são representados por uma sequência com uma relação cronológica associada à variável de indexação, sendo os sinais contínuos descritos por uma função onde essa variável é agora associada à grandeza tempo. Representaremos uma sequência complexa de comprimento ou duração N, por um vector coluna $x \in \mathbb{C}^N$, com componentes, ou amostras, $[x_0, x_1, \ldots, x_{N-1}]^T$. Quando daí não resultar qualquer confusão, usaremos alternativamente a notação x_n em vez de $x[n]$.

As componentes de um vector são designadas em processamento digital de sinal (PDS) por amostras e na TCCE por símbolos ou palavras. Neste trabalho usaremos qualquer um dos termos.

2.1 Preliminares sobre álgebra linear

Alguns resultados importantes para este trabalho sobre teoria de matrizes são revistos nesta secção tendo por base alguns livros de referência nesta área [Horn85, Bronson91, Horn91, Kincaid02].

Pressupõe-se um conhecimento dos fundamentos da teoria das matrizes, e o que se segue serve mais para fixar a notação ao longo do trabalho.

2.1.1 Notações e definições

Usa-se a notação A^T para **transposta** duma matriz, ou seja, $[A^T]_{ij} = a_{ji}$. Uma matriz A que tenha a propriedade $A = A^T$ diz-se uma matriz **simétrica**.

A matriz A^H é a **conjugada da transposta** duma matriz, $[A^H]_{ij} = a^*_{ji}$. Para matrizes reais, $A^H = A^T$. A matriz A, complexa ou não, é **hermítica** ou **hermitiana**, se for igual à sua conjugada transposta, ou seja, $A^H = A$.

Um **vector próprio** de uma matriz quadrada A, é um vector coluna v não nulo que satisfaz a equação

$$Av = \lambda v$$

para um determinado escalar λ. Este escalar designa-se por **valor próprio** de A associado ao vector próprio v. O **espectro** de uma matriz A, $\sigma(A)$, é o conjunto de todos os seus valores próprios.

Dois sistemas de equações são **equivalentes** se tiverem o mesmo conjunto de soluções, isto é, ao resolver um sistema equivalente nenhuma solução é perdida nem acrescentada. Este conceito simples é de extrema importância na resolução de sistemas de equações lineares, já que, muitos métodos transformam através de **operações elementares** a matriz do sistema numa outra equivalente com estrutura mais favorável.

Considerando que χ_i representa a i-ésima equação, as operações elementares podem ser de três tipos.

- troca de duas equações no sistema, $\chi_i \leftrightarrow \chi_j$
- multiplicação de uma equação por $\alpha \neq 0 \in \mathbb{R}$, $\alpha\, \chi_i \rightarrow \chi_i$
- adição de uma equação a um múltiplo de outra, $\chi_i + \alpha\, \chi_j \rightarrow \chi_i$.

Mais à frente, no capítulo três, voltamos a este assunto quando tratarmos de pré-condicionamento de matrizes.

Admitindo uma matriz A e os conjuntos de índices $\alpha = \{1,2,\ldots,m\}$ e $\beta = \{1,2,\ldots,n\}$, é possível obter matrizes a partir de A retirando-lhe linhas e colunas. A **sub-matriz** que se obtém cujas linhas são indexadas pelo conjunto α e as colunas por β denota-se por $A(\alpha,\beta)$.

Por exemplo

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}(\{1,3\},\{1,2,3\}) = \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}.$$

Se $m=n$ e $\alpha=\beta$ a sub-matriz designa-se por **sub-matriz principal de *A***.

O **determinante** de uma matriz quadrada escreve-se como ***det***(A).

Se A e B são duas matrizes tais que $AB=I$, onde I designa a matriz identidade, diz-se que B é a **inversa direita** de A e que A é a **inversa esquerda** de B. O exemplo

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \alpha & \beta \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

mostra que a inversa direita, caso exista, não é necessariamente única.

Para matrizes quadradas a situação é mais favorável: as inversas direita e esquerda quando existem são únicas e iguais, tendo-se $AB=BA=I$. A matriz B designa-se **inversa** de A e diz-se que a matriz A é **invertível** ou **não-singular**. É claro que B também é invertível sendo A a sua inversa. Escrevemos $B=A^{-1}$ e $A=B^{-1}$.

Se A é invertível, o sistema de equações lineares $Ax=b$ tem uma solução única, $x=A^{-1}b$. Se a inversa de A estiver disponível numericamente, esta equação representa um bom método para obter a solução. Regra geral, esta situação não se verifica e para muitos problemas o cálculo da inversa de uma matriz pode ser complicado. No capítulo seguinte abordaremos processos mais eficientes para obter soluções de sistemas de equações lineares.

TEOREMA 1, [Horn85]

- Propriedades de uma matriz não-singular -

Para uma matriz A real não-singular de dimensão $N \times N$ as seguintes propriedades são equivalentes:

- a inversa de A existe;
- ***det***(A) é diferente de zero;
- as linhas de A formam uma base de $\mathbb{R}^N$;
- as colunas de A formam uma base de $\mathbb{R}^N$;
- a equação $Ax=0$ implica $x=0$;
- para cada $b \in \mathbb{R}^N$, existe apenas um $x \in \mathbb{R}^N$ de tal forma que $Ax=b$;
- 0 não é um valor próprio da matriz A.

■

O **raio espectral** de uma matriz M, $\rho(M)$, é o maior de todos os seus valores próprios em valor absoluto,

$$\rho(M) = \max_i |\lambda_i| .$$

Se $\rho(M)<1$, então a inversa da matriz $I-M$ existe e o sistema de equações lineares $x=Mx+b$ associado à matriz tem solução. Esta pode ser obtida por um método directo com ou sem factorização, ou iterando $x_{i+1}=Mx_i+b$ [Ferreira94A].

Um conceito importante em álgebra é a noção de matriz positiva definida. Uma matriz A é **positiva definida** se a forma quadrática associada é positiva, ou seja, $x^H Ax > 0$, $\forall x \neq 0$. Ao definirmos uma forma quadrática podemos assumir sem perda de generalidade que a matriz é hermítica [Kay88].

Uma matriz A é **não-negativa definida** se a forma quadrática associada é não-negativa ou nula, ou seja, $x^H Ax \geq 0$, $\forall x \neq 0$.

2.1.2 Matrizes especiais

O símbolo de ***Kronecker*** é definido por

$$\delta[i] = \begin{cases} 1, & \text{se } i = 0 \\ 0, & \text{outros casos} \end{cases}.$$

Uma matriz quadrada complexa A é **unitária** se

$$A^{-1}=A^{H}.$$

Quando A é unitária as linhas (colunas) de A satisfazem a

$$a_i^H a_j = \delta[i-j].$$

Uma matriz unitária importante para este trabalho é a matriz de **Fourier** $N\times N$,

$$F_{mn} = \frac{1}{\sqrt{N}} e^{-\mathrm{j}\frac{2\pi}{N}mn}, \qquad m, n = 0, 1, \ldots, N\text{-}1,$$

onde $\mathbf{j}=\sqrt{-1}$ representa a unidade complexa.

Existem quatro modelos possíveis para problemas de reconstrução de sinais que correspondem a todas as combinações tempo e frequência, contínuo e discreto. Neste trabalho interessa-nos tratar de sinais de dimensão finita (ou periódicos) que são caracterizados por vectores no domínio do tempo ou da frequência. Na terminologia de [Sanz83] corresponde ao modelo discreto-discreto.

Um sinal de dimensão finita, representado pelo vector x, limitado em frequência obedece à equação $x=Bx$, onde B representa uma matriz de **limitação em frequência** de dimensão $N\times N$. Pode exprimir-se por

$$B= F^H \Gamma F. \tag{2.1}$$

O operador Γ (**amostragem na frequência)** de dimensão $N\times N$, é uma matriz diagonal contendo apenas zeros e uns que determinam a banda passante de B.

A densidade da matriz Γ pode definir-se como o cociente entre o número de elementos não nulos da sua diagonal (K) e o número de elementos dessa diagonal, N. Normalmente designa-se este parâmetro por **largura de banda normalizada**.

$$LB_{norm} = \frac{K}{N}.$$

Uma matriz **circulante** $N\times N$ é aquela onde os elementos da i-ésima linha podem ser obtidos a partir da linha (i-1) realizando uma rotação circular para a direita

$$A = \begin{bmatrix} a_0 & a_1 & \cdots & a_{N-1} \\ a_{N-1} & a_0 & \cdots & a_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & \cdots & a_0 \end{bmatrix}.$$

Quando a matriz é circulante é possível exprimi-la em função do produto

$$A = \sum_{k=0}^{N-1} a_k P^k,$$

onde P é a matriz de **permutação** dada por

$$P = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

P^0 é a matriz identidade, e por análise dos vectores e valores próprios de P é possível concluir que os valores próprios duma matriz circulante são iguais a [Kay88]

$$\lambda_i = \sum_{k=0}^{N-1} a_k e^{-\mathrm{j}\frac{2\pi}{N}k(i-1)} \text{ , para } i = 1,\ldots,N. \qquad (2.2)$$

Ou seja, calculando a transformada discreta de Fourier (DFT) dos elementos da primeira linha da matriz circulante é possível obter os seus valores próprios.

A inversa de uma matriz circulante pode ser calculada através dos seus valores próprios e da matriz unitária de Fourier da seguinte forma

$$A^{-1}=F\Lambda^{-1}F^{H}$$

onde,

$$\Lambda^{-1} = \boldsymbol{diag}\left(\frac{1}{\lambda_1},\frac{1}{\lambda_2},\cdots,\frac{1}{\lambda_N}\right)$$

e λ_i é dado por (2.2).

Uma matriz $N\times N$ **idempotente** (**reprodutora**) satisfaz a

$$A^2=A.$$

A matriz B de limitação em frequência do tipo (2.1) é diagonalizável, real, simétrica, circulante, não-negativa definida, idempotente, possui K valores próprios unitários e N-K valores próprios nulos [Ferreira94B]. A matriz terá um conjunto de N-K valores próprios nulos contíguos, se o conjunto formado pelos índices da diagonal de Γ também o for [Ferreira94C]. A transformada de Fourier da sua primeira linha tem k amostras nulas, contíguas módulo-N [Ferreira94A]. Qualquer matriz de limitação em frequência, ou de forma equivalente, amostragem no domínio da frequência, é uma matriz de **projecção,** ou seja, idempotente [Ferreira94C].

Uma matriz ***Toeplitz*** de dimensão $N\times N$ pode definir-se por

$$[A]_{ij} = a_{i-j},$$

ou

$$A = \begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & a_{-(N-1)} \\ a_1 & a_0 & a_{-1} & \cdots & a_{-(N-2)} \\ a_2 & a_1 & a_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & a_{-1} \\ a_{N-1} & a_{N-2} & \cdots & a_1 & a_0 \end{bmatrix}.$$

A simetria deste tipo de matrizes em relação à diagonal principal é, em relação aos pontos cardeais, de NW-SE. Se $a_{-k} = a_k^*$, a matriz designa-se ***Toeplitz*** **hermítica**. Se a matriz for real e $a_{-k} = a_k$ é ***Toeplitz*** **simétrica**.

Se a matriz for real a sua inversa não é Toeplitz, mas é simétrica em relação às duas diagonais: a principal e a secundária. Para matrizes complexas, a inversa de uma matriz Hermítica *Toeplitz* é Hermítica e simétrica em relação à diagonal secundária, ou seja, é uma matriz **persimétrica**. Pode definir-se A^{-1}

$$[A^{-1}]_{ij} = [A^{-1}]_{N-j+1,N-i+1}.$$

Se A for uma matriz Hermítica *Toeplitz*, 4×4 então

$$A^{-1} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{12}^* & b_{22} & b_{23} & b_{13} \\ b_{13}^* & b_{23}^* & b_{22} & b_{12} \\ b_{14}^* & b_{13}^* & b_{12}^* & b_{11} \end{bmatrix},$$

e existe uma simetria em relação à diagonal secundária de NE-SW.

A estrutura da matriz influencia a pesquisa da solução do sistema de equações lineares $Ax=b$. Na secção 3.4 voltamos ao assunto quando nos referirmos ao pré-condicionamento de matrizes e estabilidade associada a códigos que usam a transformada de Fourier, já que se podem estabelecer relações entre a estrutura e os valores próprios dessa matriz e o padrão de amostras perdidas.

2.1.3 Normas e medidas de erros

Quando se discutem erros em problemas de análise numérica com vectores, é imprescindível a utilização de normas. Considere-se um processo iterativo que procura um número real que é um zero de uma equação complicada, ou um valor numérico de

um integral definido que não se obtém directamente. A sequência obtida pode convergir para x

$$\lim_{n\to\infty} x_n = x\,,$$

mas a convergência pode apresentar **comportamentos distintos** [Kincaid02]. De um modo geral, se existirem duas constantes positivas C e α e um inteiro N de tal forma que

$$|x_{n+1} - x| \le C|x_n - x|^{\alpha}, \quad (n \ge N),$$

a taxa de convergência é **geométrica** de **ordem** α, ou seja, $\vartheta(\alpha)$. A convergência dos algoritmos de reconstrução é por vezes geométrica.

Gostaríamos de dispor de conceitos análogos para algoritmos de reconstrução, mas estes são formulados em $\mathbb{R}^N$ ($\mathbb{C}^N$). O papel realizado pela função $|x|$, que se pode entender como uma medida do "tamanho" do real x, ou da sua distância à origem é desempenhado em $\mathbb{R}^N$ ($\mathbb{C}^N$) por funções $\mathbb{R}^N \to \mathbb{R}_0^+$ ou $\mathbb{C}^N \to \mathbb{R}_0^+$, as normas.

NORMA VECTORIAL.

Num espaço vectorial V, uma norma é uma função $\|\cdot\|$ de V para um conjunto de reais não negativos que satisfaz a:

- $\|x\| \ge 0$ se $x \ne 0, x \in V$, (2.3)
- $\|\alpha x\| = |\alpha| \cdot \|x\|$ se $\alpha \in \mathbb{R}, x \in V$, (2.4)
- $\|x + y\| \le \|x\| + \|y\|$ se $x, y \in V$. (2.5)

Podemos entender a função $\|x\|$ como sendo o tamanho ou comprimento do vector x. Num espaço vectorial a norma de um vector $\|\cdot\|$ generaliza o conceito de valor absoluto $|\cdot|$. Assim, é possível definir "distância" entre dois elementos de um espaço normado

$$d(a,b) = \|a - b\|,$$

ou seja, $d(.)$ é a distância induzida pela norma $\|\cdot\|$.

A norma mais usual em $\mathbb{R}^N$ é a norma Euclideana, a que mais se aproxima ao conceito intuitivo de comprimento de um vector,

$$\|x\|_e = \sqrt{\sum_{i=1}^{N} x_i^2}\ , \quad x = [x_1, x_2, \ldots, x_N]^T .$$

Para vectores existem normas alternativas designadas por normas l_p

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^{N} |x_i|^p} .$$

Entre estas, as mais utilizadas são as

$$\|x\|_\infty = \max_{1\le i\le N} |x_i| \quad \text{e} \qquad \|x\|_1 = \sum_{i=1}^{N} |x_i| ,$$

l_∞ e l_1 respectivamente. De notar que para vectores $\|x\|_2 = \|x\|_e$.

NORMA MATRICIAL.

Podem estabelecer-se funções norma matricial a partir dos mesmos pressupostos que indicamos para a norma de vectores (2.3 a 2.5). No entanto, é para nós de maior utilidade definir uma norma matricial relacionada com a norma vectorial, o que está mais de acordo com a interpretação da matriz como operador linear.

DEFINIÇÃO 1, [Kincaid02]

- Norma matricial induzida por uma vectorial -

Admita-se uma matriz A de dimensão $N\times N$, e uma função $\|\cdot\|$ norma em $\mathbb{R}^N$. A equação

$$\|A\| = \sup_{\|u\|=1} \left\{ \|Au\| : u \in \mathbb{R}^N \right\}$$

define a norma no espaço linear de todas as matrizes $N\times N$.

■

Desta definição surge uma relação importante,

$$\|Ax\| \le \|A\| \|x\| \quad (x \in \mathbb{R}^N). \tag{2.6}$$

Esta expressão é verdadeira para $x=0$. Se $x \neq 0$, o vector $v=x / \|x\|$ tem norma unitária. Assim, temos

$$\|A\| \ge \|Av\| = \frac{\|Ax\|}{\|x\|} .$$

Para ilustrar este conceito, vamos calcular a norma matricial induzida pela vectorial l_∞.

$$\|A\|_\infty = \sup_{\|u\|_\infty=1} \|Au\|_\infty$$

$$= \sup_{\|u\|_\infty=1} \left\{ \max_{1\le i\le N} |(Au)_i| \right\} = \max_{1\le i\le N} \left\{ \sup_{\|u\|_\infty=1} |(Au)_i| \right\}$$

$$= \max_{1\le i\le N} \left\{ \sup_{\|u\|_\infty=1} \left| \sum_{j=1}^{N} a_{ij} u_j \right| \right\} = \max_{1\le i\le N} \sum_{j=i}^{N} |a_{ij}| .$$

Considerou-se que o supremo de $\left|\sum_{j=1}^{N} a_{ij} u_j\right|$ para um i fixo e $\|u\|_\infty = 1$, é obtido colocando u_j=+1, se $a_{ij} \ge 0$ e u_j=-1, se $a_{ij} < 0$ [Kincaid02].

Uma função norma matricial induzida por uma vectorial tem propriedades adicionais àquelas definidas por 2.3 a 2.5. Assim a partir da definição 1 e de (2.6) podemos escrever

$$\| AB\| \le \| A \| \|B \| .$$

2.1.4 Estruturas oblíquas em espaços de dimensão finita

Muitos problemas importantes de processamento de sinal são formulados de uma forma natural em espaços de Hilbert. O produto interno é definido por

$$\langle x, y \rangle = y^H x = \sum_{i=1}^{N} x_i y_i^* ,$$

e a norma induzida por este produto interno é

$$\|x\| = \sqrt{\langle x, x \rangle} .$$

Assim temos

$$\|x\|^2 = x^H x = \sum_{i=1}^{N} |x_i|^2 .$$

Considere-se a matriz G=[g_1, g_2, …, g_N] de dimensão N×N não singular, que pelo teorema 1 possui colunas linearmente independentes. Estas colunas $\{g_i\}$ formam uma **base** do espaço $\mathbb{C}^N$, que é um espaço de Hilbert (H) de dimensão finita [Kaiser94, Cadzow97].

Qualquer vector $x \in \mathbb{C}^N$ pode ser expresso sob a forma

$$x = \sum_{i=1}^{N} \sigma_i g_i$$

para uma sequência única de escalares σ_i. Em notação matricial podemos escrever x=$G\sigma$, onde $\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_N]^T$. A matriz G é uma transformação de $\mathbb{C}^N$ em $\mathbb{C}^N$ e o

facto de existir inversa para G, faz com que qualquer valor $x \in \mathbb{C}^N$ possa ser determinado por um único σ tal que, $x=G\sigma$.

Admitindo uma situação prática onde se pretende obter x a partir de σ, o ideal será que sendo a norma $\| \sigma \|$ pequena, então a norma de $\| x \|$ também deve ser, e vice-versa. Esta situação corresponde à existência de um erro pequeno para x quando ocorre um pequeno erro na estimação ou transmissão de σ.

Sendo $x=G\sigma$, é clara a dependência entre o que foi referido no parágrafo anterior e a norma de $\| G \|$ que pode ser caracterizada em termos dos seus valores singulares ou próprios. Sendo $x = \sum_{i=1}^{N} \sigma_i g_i$ e $\|\sigma\|^2 = \sum_{i=1}^{N} |\sigma_i|^2$ é possível relacionar a norma $\| x \|^2$ com $\| \sigma \|^2$.

Demonstra-se em [Chui92] que qualquer base $\{g_i\}$ dum espaço de dimensão finita satisfaz a

$$\alpha \sum_{i=1}^{N} |c_i|^2 \le \left\| \sum_{i=1}^{N} c_i g_i \right\|^2 \le \beta \sum_{i=1}^{N} |c_i|^2 ,$$

onde $\alpha=\lambda_m>0$ e $\beta=\lambda_M<\infty$ representam o menor e maior valor próprio de G respectivamente. Está assim assegurada a estabilidade entre as transformações de x para $\{c_i\}$ e de $\{c_i\}$ para x.

O conceito de estruturas oblíquas (não ortogonal) ou *frames*, foi introduzido por Duffin e Schaeffer [Duffin52] e voltou à actualidade no contexto das ôndulas (*wavelets*) por vários autores nomeadamente Daubechies [Daubechies92].

Uma estrutura oblíqua não constitui forçosamente uma base, mas tem propriedades semelhantes às bases. Por exemplo podemos escrever um vector do espaço de Hilbert $x \in \mathbb{H}$ como uma combinação linear dos vectores que constituem a estrutura oblíqua $\{g_i\}$, ou seja, $x = \sum_{i=1}^{N} c_i g_i$. Em geral, e ao contrário das bases, os vectores que constituem a estrutura não são linearmente independentes, ou seja, existe redundância na estrutura oblíqua.

Interessa para este trabalho definir *frames* finitas dada a sua simplicidade e aplicabilidade prática. Um trabalho recente sobre o tema pode encontrar-se em [Pei97].

DEFINIÇÃO 2

- Estruturas Oblíquas Finitas -

Uma sequência de vectores $\{g_i\}$ pertencentes a um espaço de Hilbert $\mathbb{H}$ constitui uma estrutura oblíqua se existirem constantes α e β, com $0<\alpha\leq\beta<\infty$ tais que, para qualquer $x\in\mathbb{H}$

$$\alpha\|x\|^2 \leq \sum_{i=1}^{N}\left|\langle x, g_i\rangle\right|^2 \leq \beta\|x\|^2 .$$

As constantes α e β, designam-se por limites da estrutura oblíqua.

■

Os limites das *frames* são importantes porque determinam até que ponto as imprecisões nos dados se propagam à solução do problema inverso que corresponde à reconstrução. Mais à frente voltaremos a este assunto quando abordarmos o tema da estabilidade no contexto de *frames* no capítulo cinco.

2.2 Aritmética de computadores e problemas de instabilidade numérica

O produto de dois números decimais x com d dígitos à direita da vírgula é um número com $2d$ casas decimais. Sendo fixa a quantidade de dígitos disponíveis, é necessário arredondar o resultado podendo esta operação realizar-se de várias formas.

Considerando $\tilde{x}$ uma aproximação de x com n-dígitos obtida por arredondamento, temos a seguinte desigualdade

$$\left|x-\tilde{x}\right| \leq \tfrac{1}{2}\times 10^{-n} .$$

Se o dígito $(n+1)$ de x for 0, 1, 2, 3 ou 4, $x=\tilde{x}+\varepsilon$ com $\varepsilon<\frac{1}{2}\times 10^{-n}$. Se for 5, 6, 7, 8 ou 9, $\tilde{x}=\hat{x}+10^{-n}$ onde $\hat{x}$ é um número com os n dígitos iguais aos de x e todos os outros iguais a zero. Assim, $x=\hat{x}+\delta\times 10^{-n}$ com $\delta\geq\frac{1}{2}$ o que faz com que $\tilde{x}-x=(1-\delta)\times 10^{-n}$.

Truncar pode ser outro processo de aproximar dois números. Neste caso, o número $\hat{x}$ é obtido de uma forma simples desprezando todos os (n-d) dígitos à direita do dígito n, ou seja,

$$\left|x-\hat{x}\right| \leq 10^{-n} .$$

Nesta situação, $x=\hat{x}+\delta\times 10^{-n}$ com $0\leq\delta<1$.

A necessidade de comunicação entre computadores com diferentes concepções, tanto ao nível do hardware como de software, tornou inevitável a criação de standards nomeadamente no que diz respeito à representação dos dados. A norma IEEE [ANSI85] especifica a aritmética e representação de vírgula flutuante em precisão **simples e dupla** (32 e 64 bits respectivamente) usada em muitos computadores.

Se admitirmos uma aritmética de 32-bits, a representação de um número em precisão simples é descrita na figura 2.1.

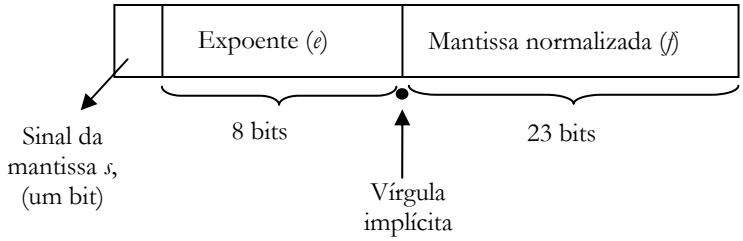

Figura 2.1: Representação de um número máquina em precisão simples.

Nos pontos seguintes, quando nos referimos a um número máquina admite-se uma aritmética de 32-bits para a sua representação.

Qualquer número diferente de zero é nesta representação uma palavra descrita pela expressão

$$x = (-1)^s q \times 2^k, \tag{2.7}$$

onde

$$q=(1.f)_2 \quad \text{e} \quad k=e-127.$$

O bit mais significativo de q é 1, não é explicitamente guardado: sendo assim, $1 \le q < 2$. O sinal de x é descrito por um bit: se for um número positivo, $s = 0$ e caso seja negativo $s = 1$. São reservados 8 bits para o expoente k e f é uma palavra de 23 bits que representa a parte decimal do número real x admitindo-se que o primeiro bit seja 1. Temos então um número no sistema binário da forma $(1.\square\square\square\ldots\square\square\square)_2$.

A restrição de 8 bits imposta a $|k|$, aliada à necessidade de reservar $e=0$ e $e=255$ para casos especiais como ± 0, $\pm\infty$ e NAN, faz com que $-126 \le k \le 127$. Assim, com 32 bits os números mais pequeno e maior que se conseguem representar em valor absoluto são respectivamente $2^{-126} \approx 1.2\times10^{-38}$ e $[(2-2^{23})2^{127}] \approx 3.4\times10^{38}$.

Um número real descrito pela expressão (2.7) diz-se normalizado em vírgula flutuante e será um número máquina se puder ser representado neste computador sem erros.

Apesar da norma [ANSI85] usar 80 bits em cálculos internos, na prática a maior parte dos números reais são aproximados ao número máquina mais próximo dada a gama finita de representação disponível.

Pode acontecer, no decurso dum cálculo computacional que um número seja produzido fora da gama permitida pelo computador. Se k <-126 ou k > 127, diz-se que ocorreu *underflow* ou *overflow*, respectivamente.

Para representar números inteiros, com excepção do bit de sinal, podem usar-se todos os 31 bits o que equivale a ter inteiros compreendidos entre $-(2^{31}-1)$ e $+(2^{31}-1)=2147483647$.

Em muitos casos utilizar uma palavra de 32 bits para representar números não é suficiente. A representação de vírgula flutuante em precisão dupla é descrita por duas palavras, ou seja, 64 bits. Como a mantissa normalmente tem o dobro dos bits, existe também o dobro de casas decimais de precisão. Este factor contribui decisivamente para que toda a aritmética seja mais lenta (e o custo do cálculo seja incrementado) por um factor que varia de 2 a 4 [Kincaid02].

2.2.1 Erro relativo e erro absoluto. Perda de significância nos dados

Podemos calcular os limites para o erro quando aproximamos um número real positivo x ao número máquina mais próximo. Admitindo

$$x = q \times 2^k \qquad 1 \le q < 2 \qquad -126 \le k \le 127$$

se

$$x = \left(1.d_1 d_2 \ldots d_{23} d_{24} d_{25} \ldots\right)_2 \times 2^k$$

o número máquina mais próximo à esquerda de x é

$$x_- = \left(1.d_1 d_2 \ldots d_{23}\right)_2 \times 2^k$$

e x_+ é o que está mais próximo à direita

$$x_+ = \left(\left(1.d_1 d_2 \ldots d_{23}\right) + 2^{-23}\right) \times 2^k .$$

O valor para o qual a distância em relação a x for menor, será o escolhido para representar x no computador. Designando este valor por $\tilde{x}$ e se $\tilde{x} = x_-$ temos

$$\left|x - \tilde{x}\right| \le \tfrac{1}{2}\left|x_+ - x_-\right| = \tfrac{1}{2} \times 2^{k-23} = 2^{k-24} .$$

O **erro absoluto** é dado por

$$\left|x - \tilde{x}\right|$$

e o **erro relativo** tem como limite superior

$$\left|\frac{x-\tilde{x}}{x}\right| \le \frac{2^{k-24}}{q\times 2^{k}} = \frac{1}{q}\times 2^{-24} \le 2^{-24}.$$

Considerando $\delta = (\tilde{x}-x)/x$, podemos escrever

$$fl(x) = x(1+\delta) \qquad |\delta| \le 2^{-24}.$$

A notação *fl* (*x*) é usada para representar o número máquina $\tilde{x}$ mais próximo de *x*. O número 2^{-24} designa-se por unidade de erro de arredondamento. Podemos concluir assim que a quantidade de bits associados à mantissa, está directamente relacionada com a unidade de erro de arredondamento e determina a precisão da aritmética do computador.

Genericamente, numa máquina que utilize a base *b* e possua *n* casas decimais na mantissa na sua representação em vírgula flutuante, temos

$$fl(x) = x(1+\delta) \quad , \quad |\delta| \le \varepsilon$$

sendo $\varepsilon = \frac{1}{2}b^{1-n}$ quando se arredonda e $\varepsilon = b^{1-n}$ quando se trunca.

TEOREMA 2, [Kincaid02]

- Erro relativo de arredondamento numa adição -

Seja $x_0, x_1, \ldots, x_n$ um conjunto de números máquina positivos cuja unidade de erro de arredondamento é ε. O erro relativo de arredondamento quando se adicionam *n* números

$$\sum_{i=0}^{n} x_i$$

não excede $(1+\varepsilon)^n - 1 \approx n\varepsilon$.

■

A unidade de arredondamento ε é característica intrínseca de cada computador. O seu valor depende do tipo de arquitectura, do tamanho da palavra usada para representar os dados e do tipo de arredondamento utilizado (alguns computadores usam aritméticas não binárias, hexadecimal ou octal por exemplo, utilizando outro tipo de arredondamentos). A possibilidade que alguns compiladores oferecem de seleccionar o tipo de arredondamento ou de optar por precisão simples ou dupla em algumas situações permite adaptar o código ao hardware no sentido de aumentar por exemplo o número de MFLOPS [Frigo99].

Apesar dos erros de arredondamento serem inevitáveis e difíceis de controlar, outros existem que com cuidado na programação podem ser evitados. A subtracção de dois números muito próximos ($x - y$) pode originar erros relativos grandes ou seja **perda de significância nos dados**. O número de bits significativos que se perdem depende do valor particular de x e y. O teorema seguinte estabelece limites para a quantidade $\left|1 - \frac{y}{x}\right|$ que é uma medida da proximidade dos dois valores x e y.

TEOREMA 3, [Kincaid02]

- Perda de precisão -

Se x e y são dois números máquina positivos normalizados em virgula flutuante binária de tal forma que $x > y$ e

$$2^{-q} \leq 1 - \frac{y}{x} \leq 2^{-p},$$

então no máximo q e no mínimo p bits significativos são perdidos na subtracção $x–y$.

■

Em alguns cálculos, a perda de significância pode ser atenuada se utilizarmos precisão dupla com as desvantagens inerentes associadas ao custo de execução. Recentes trabalhos usam conceitos como aritmética *fuzzy* ou intervalo aritmético para controlar a extensão dos erros de arredondamento no cálculo computacional [Klir95]. Cada número calculado é acompanhado dum intervalo onde garantidamente se encontra o seu valor exacto. O custo de transportar este intervalo ao longo dos cálculos é óbvio à medida que o número de operações ou a ordem do sistema aumentam. Estes factores fazem com que estes métodos se usem apenas quando se pretende grande confiança nos dados. Neste trabalho não serão usados intervalos aritméticos.

2.2.2 Erros em sistemas de equações lineares e número de condição

Um processo numérico é **instável** se erros pequenos que ocorram num determinado estágio de um processo são amplificados nos passos seguintes de tal forma que a precisão total dos dados é seriamente afectada.

A noção de erro ou limites para o erro está intimamente ligada ao conceito de distância, ou seja, podemos usar a medida de distância entre dois vectores para quantificar o erro entre eles. Vamos utilizar normas vectoriais e de matrizes já que vimos

no ponto 2.1.3 que uma distância induz uma norma e vice-versa. Assim, o erro final é dado por $\|x-y\|_p$ onde x representa o sinal original e y o sinal obtido no final do processo de reconstrução. O erro residual, que geralmente serve de base ao critério de paragem utilizado nos métodos iterativos, é dado por $\left\|y^{(i+1)}-y^{(i)}\right\|_p$, onde $y^{(i)}$ representa aproximação ao sinal original na iteração-*i*. As normas tradicionais e que vão ser mais usadas ao longo do trabalho, são as correspondentes a p=2 e p=∞. Para uma análise mais detalhada pode consultar-se [Horn85, Chapra88]

A utilização destas normas como estratégia para avaliar o comportamento dos algoritmos de resolução de equações lineares, sendo a mais usual e a utilizada neste trabalho, não é única. Em [Tarczynsky97, Wang99] são propostas metodologias alternativas para a qualidade da reconstrução.

Na obtenção da solução de um problema genérico, a indicação do grau de sensibilidade no seu cálculo quando existem pequenas variações relativas nos dados de entrada, pode definir-se como **condicionamento** do problema. Este será mal condicionado quando pequenas variações nos dados alteram dramaticamente os valores esperados. Para alguns problemas, pode definir-se um número de condição como medida da sensibilidade definida anteriormente. Se o seu valor é elevado, o problema é extremamente sensível logo mal condicionado, e vice-versa.

Apesar do número de condição poder ser útil em vários contextos, interessa sob o ponto de vista deste trabalho defini-lo no âmbito da análise numérica, mais propriamente na resolução de equações lineares, já que como veremos no capítulo cinco o problema da reconstrução vai consistir na formulação de dois tipos de sistemas de equações lineares.

Admita-se então um sistema desse tipo $Ax=b$ onde A é uma matriz invertível. Supondo que ocorre uma perturbação em A^{-1}, originada por um erro de representação por exemplo, a solução $x=A^{-1}b$ passa a ser o vector $\tilde{x}=A^{\gamma}b$ onde a matriz A^{γ} representa uma aproximação à inversa de A. Podemos ter uma medida do valor absoluto da perturbação, ou seja,

$$\|x-\tilde{x}\|=\left\|x-A^{\gamma}b\right\|=\left\|x-A^{\gamma}Ax\right\|=\left\|\left(I-A^{\gamma}A\right)x\right\|\leq\left\|I-A^{\gamma}A\right\|\|x\|.$$

O limite superior da inequação seguinte pode ser usado para obter um majorante para o erro relativo entre x e $\widetilde{x}$

$$\frac{\|x-\widetilde{x}\|}{\|x\|} \le \|I - A^{\gamma} A\| .$$

Podemos relacionar o número de condição com a precisão absoluta ou relativa nos dados quando existe perturbação no vector *b*, senão vejamos: sendo $\widetilde{b} = fl(b) \neq b$ a solução obtida é regra geral $\widetilde{x} \neq x$, logo

$$A\widetilde{x} = \widetilde{b} .$$

Supondo que existe A^{-1}, podemos escrever

$$\begin{aligned} A(x-\widetilde{x}) &= (b-\widetilde{b}) \\ x-\widetilde{x} &= A^{-1}(b-\widetilde{b}) \\ \|x-\widetilde{x}\| &= \|A^{-1}(b-\widetilde{b})\| \\ &\le \|A^{-1}\|\|b-\widetilde{b}\| . \end{aligned}$$

Esta expressão dá-nos uma medida da perturbação em *x*.

Se pretendermos estimar a perturbação relativa, temos

$$\|x-\widetilde{x}\| \le \|Ax\|\|A^{-1}\|\frac{\|b-\widetilde{b}\|}{\|b\|} \le \|A\|\|x\|\|A^{-1}\|\frac{\|b-\widetilde{b}\|}{\|b\|},$$

$$\frac{\|x-\widetilde{x}\|}{\|x\|} \le \|A\|\|A^{-1}\|\frac{\|b-\widetilde{b}\|}{\|b\|} .$$

O produto das normas da matriz e da sua inversa,

$$\kappa(A) = \|A\|\|A^{-1}\|$$

é o número de condição da matriz A [Kincaid02]. Este número é sempre positivo e satisfaz a $\kappa(A) \ge 1$, porque

$$\kappa(A) = \|A\|\|A^{-1}\| \ge \|AA^{-1}\| = \|I\| = 1,$$

para normas induzidas (2.1.3).

Assim, podemos exprimir o erro relativo em função desse valor

$$\frac{\|x-\widetilde{x}\|}{\|x\|} \le \kappa(A)\frac{\|b-\widetilde{b}\|}{\|b\|} . \qquad (2.8)$$

O número de condição depende da norma matricial escolhida. No entanto, em espaços de dimensão finita tem-se a seguinte desigualdade que relaciona duas normas l_p e l_q ($p \neq q$) [Horn85]:

$$\alpha \|x\|_q \leq \|x\|_p \leq \beta \|x\|_q \quad \alpha, \beta \text{ constantes } \alpha \leq \beta \cdot$$

Quando resolvemos numericamente o sistema de equações

$$Ax = b$$

a solução exacta é impossível de obter devido, entre outros, a erros de representação. A solução obtida, $\tilde{x}$, deve estar o mais próximo de x. O vector residual r dá-nos uma medida da proximidade entre b e $A\tilde{x}$

$$r = b - A\tilde{x} .$$

A diferença entre a solução exacta e a aproximação $\tilde{x}$ denomina-se vector erro

$$e = x - \tilde{x} .$$

Logo podemos escrever

$$Ae = r$$

e verificar que o efeito de aplicar a matriz A à perturbação sentida em x é igual à perturbação manifestada em b.

Partindo novamente da inequação (2.8) podemos agora estabelecer uma relação entre o número de condição $\kappa(A)$, o vector residual r e o vector de erro e. Assim, podemos escrever

$$\frac{\|e\|}{\|x\|} \leq \kappa(A) \frac{\|r\|}{\|b\|} . \tag{2.9}$$

Por outro lado temos

$$\|Ae\| \|A^{-1} b\| \leq \|A\| \|e\| \|A^{-1}\| \|b\|$$

$$\|r\| \|x\| \leq \|A\| \|A^{-1}\| \|b\| \|e\| ,$$

logo

$$\frac{1}{\kappa(A)} \frac{\|r\|}{\|b\|} \leq \frac{\|e\|}{\|x\|} . \tag{2.10}$$

Conjugando as inequações (2.9) e (2.10) podem estabelecer-se limites que relacionam os erros relativos em x e b, o número de condição, os vectores residual e erro, ou seja,

$$\frac{1}{\kappa(A)} \frac{\|r\|}{\|b\|} \leq \frac{\|e\|}{\|x\|} \leq \kappa(A) \frac{\|r\|}{\|b\|} .$$

A matriz A é bem condicionada quando o número $\kappa(A)$ apresenta valores modestos, próximos da unidade. Quando este valor é elevado, diz-se que a matriz A é mal condicionada. Neste caso, para manter uma determinada precisão no cálculo de x é necessário que a precisão correspondente em b seja superior em várias ordens de grandeza. Em casos extremos a solução numérica para $Ax=b$ deve ser aceite com muitas reservas. Na verdade, se $\kappa(A)\approx 10^d$, então cerca de d dígitos da solução poderão ser questionáveis.

CAPÍTULO 3

Resolução de equações lineares

A resolução de sistemas de equações lineares através de métodos iterativos e directos está em constante desenvolvimento. O aparecimento de novos métodos por um lado, e de diferentes abordagens aos já existentes por outro, reforçam esta afirmação [Gröchenig93B, Ferreira94B]. Para além disto, a possibilidade de dispormos de processadores com elevadas performances [Steen94] aliada às restrições de tempo real que muitas aplicações impõem, seleccionam naturalmente uns métodos e tornam impraticáveis outros quando pretendemos fazer uma aplicação prática.

Resolver um sistema de equações através de um método iterativo consiste genericamente em aproximar a solução do problema passo a passo até se atingir uma determinada precisão numérica. Existem muitos algoritmos incluídos em bibliotecas de software (LINPACK, LAPACK, BLAS, IML++) e o número de trabalhos sobre o assunto é grande [Young71, Dongarra79, Jones86, Demmel89, Golub89, Bronson91, Press92, Barret94, Shewchuk94, Kincaid02]. Em muitas situações, uma funcionalidade inadequada, uma estrutura de dados que não se ajusta a um determinado problema, ou a possibilidade de distribuir partes do algoritmo num multiprocessador podem obrigar à adaptação do código.

Sob outro ponto de vista, podemos para uma determinada aplicação tentar encontrar o método que melhor se ajuste não só às características do problema, mas também a factores externos como seja por exemplo a arquitectura que vai ser utilizada.

Existem muitos métodos para obter a solução destes problemas iterativamente ou de uma forma directa. Em seguida descrevemos alguns incidindo mais no seu desempenho computacional no contexto de problemas de reconstrução e descodificação de códigos reais com apagamentos e erros.

3.1 Métodos directos

O sistema de n equações e n incógnitas $x_1, x_2, \ldots, x_n$ da forma

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \qquad\qquad\qquad\qquad\qquad\quad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

com os elementos $b_i, a_{ij} \in \mathbb{R}$ pode ser descrito matricialmente por

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

ou seja,

$$Ax=b. \tag{3.1}$$

A solução é trivial quando A é diagonal, isto é

$$\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix},$$

caso em que a solução é

$$x = \begin{bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ \vdots \\ b_n/a_{nn} \end{bmatrix},$$

se A for não singular.

Se $a_{ii}=0$ e $b_i=0$ para algum índice, então x_i pode ser um número real qualquer. Se $a_{ii}=0$ e $b_i\neq 0$ não existe nenhuma solução para o sistema.

Continuando a analisar soluções para o sistema (3.1) que podem ser calculadas de uma forma simples e intuitiva, temos os casos em que a matriz A é ou pode ser transformada numa estrutura triangular inferior ou superior através de operações elementares: todos os valores diferentes de zero da matriz encontram-se para baixo ou para cima da diagonal principal, respectivamente.

Quando a matriz é triangular inferior temos

$$\begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Assumindo que $a_{ii} \neq 0 \ \forall i \in \mathbb{N}$, podemos obter x_1 a partir da primeira equação. Tendo este valor, por **substituição para a frente** (por vezes também designada por **substituição directa**) podemos encontrar todos os outros [Pina95]. Este processo pode ser descrito pelo algoritmo seguinte, onde se assume que qualquer soma do tipo $\sum_{i=\alpha}^{\beta} x_i$ quando $\beta < \alpha$, é igual a zero:

input n, (a_{ij}), (b_i)
for i=1 **to** n **do**

$$x_i \leftarrow \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j\right) \Big/ a_{ii}$$

end do
output (x_i).

Quando a matriz é triangular superior temos

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Aplicando as mesmas ideias e assumindo outra vez que $a_{ii} \neq 0 \ \forall i \in \mathbb{N}$, podemos encontrar a solução do sistema aplicando o algoritmo que normalmente se designa por **substituição para trás** (por vezes também designada por **substituição inversa**):

input n, (a_{ij}), (b_i)
for i= n **to** 1 **step** -1 **do**

$$x_i \leftarrow \left(b_i - \sum_{j=i+1}^{n} a_{ij}x_j\right) \Big/ a_{ii}$$

end do
output (x_i).

Na maior parte dos casos a matriz não tem uma estrutura assim tão simples mas a possibilidade de ser factorizada, apesar de envolver processamento adicional, conduz a bons resultados em muitas situações. A decomposição da matriz A num produto de duas matrizes sendo uma triangular inferior L e outra triangular superior U, decomposição LU, ou o caso particular deste que acontece quando $U=L^T$, designado por factorização de Cholesky, enquadram-se neste contexto.

3.1.1 Factorização *LU*

De uma forma simples, obter a solução do sistema (3.1) quando $A=LU$ compreende dois passos fundamentais:

$$\begin{array}{ll} Lz=b & \text{resolve } z \\ Ux=z & \text{resolve } x. \end{array}$$

No parágrafo anterior verificámos como é intuitiva a resolução destes dois sistemas triangulares.

O código seguinte descreve a factorização LU genérica,

input n, (a_{ij})
for k = 1 **to** n **do**

$l_{kk} \neq 0$ *solve* u_{kk} ($u_{kk} \neq 0$ *solve* l_{kk}) *by*

$$l_{kk}u_{kk} = a_{kk} - \sum_{s=1}^{k-1} l_{ks}u_{sk}$$

for j= k + 1 **to** n **do**

$$u_{kj} \leftarrow \left(a_{kj} - \sum_{s=1}^{k-1} l_{ks}u_{sj}\right) \Big/ l_{kk}$$

end do

for i= k + 1 **to** n **do**

$$l_{ik} \leftarrow \left(a_{ik} - \sum_{s=1}^{k-1} l_{is}u_{sk}\right) \Big/ u_{kk}$$

end do
end do
output (l_{ij}), (u_{ij}).

Este algoritmo designa-se por factorização de Doolittle e de Crout quando a matriz L é triangular inferior unitária (l_{ii} =1 para 1≤ $i \leq n$) e a matriz U é triangular superior unitária (u_{ii} =1 para 1≤ $i \leq n$), respectivamente. Quando $U=L^T$ de tal forma que $l_{ii} = u_{ii}$ para 1≤ $i \leq n$ a decomposição denomina-se de Cholesky [Kincaid02], como consideraremos a seguir.

Numa representação tempo/erro dum algoritmo que resolva $Ax=b$ usando esta decomposição, verifica-se que a grande fatia é gasta na obtenção de L e U, figura 3.1.

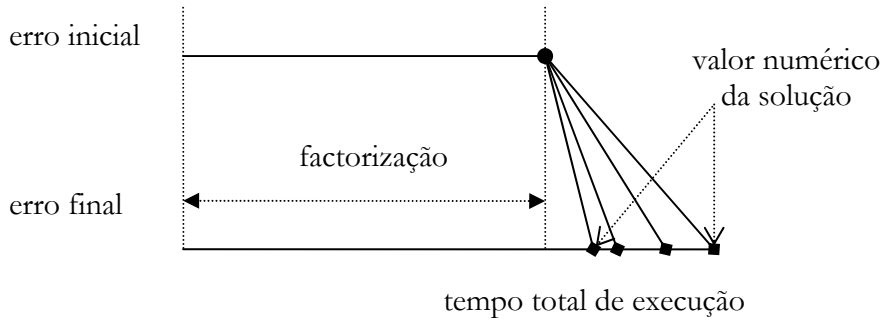

Figura 3.1: Diagrama temporal na execução dum algoritmo baseado na decomposição *LU*.

Uma nota acerca da interpretação desta representação: é claro que o erro não decresce linearmente durante a fase de substituição, que se segue à de decomposição. Na verdade a interrupção de qualquer dos processos não conduz em geral a resultados úteis. Contudo, o diagrama sugere de forma adequada a percentagem de tempo gasta na factorização que inevitavelmente só é necessária uma vez por cada matriz. O diagrama também é útil para comparação com diagramas análogos que se podem obter para os métodos iterativos.

O algoritmo tem complexidade $\vartheta(n^3)$ sendo n a dimensão do problema. A reutilização das matrizes L e U para tantos valores de b quantos se pretender pode constituir uma mais valia deste método.

3.1.2 Factorização de Cholesky

Para se decompor uma matriz A utilizando o método de Cholesky, é necessário que A tenha algumas propriedades especiais, nomeadamente: ser real, simétrica e positiva definida. Neste caso, $A=LL^T$ sendo L a matriz triangular inferior com os elementos da diagonal todos positivos.

O algoritmo para a factorização de Cholesky é um caso especial do algoritmo da factorização *LU* e pode ser descrito da seguinte forma:

input n, (a_{ij})
for $k = 1$ **to** n **do**
    $l_{kk} \leftarrow \left(a_{kk} - \sum_{s=1}^{k-1} l_{ks}^2\right)^{1/2}$
    for $i = k + 1$ **to** n **do**
    $l_{ik} \leftarrow \left(a_{ik} - \sum_{s=1}^{k-1} l_{is} l_{ks}\right) / l_{kk}$
    end do
end do
output (l_{ij}).

Para evitar o crescimento dos erros de arredondamento, tanto na factorização de Cholesky como na de Doolittle, os produtos internos vectoriais devem ser efectuados em precisão dupla.

A complexidade é cúbica, $\vartheta(n^3)$, mas para as matrizes para as quais o método é aplicável o tempo de resolução pode ser reduzido em 50% quando comparado com outras decomposições alternativas, nomeadamente *LU* [Demmel89, Press92].

Se $x^{(0)}$ for uma aproximação à solução x do sistema obtida fazendo uma decomposição *LU* ou por Cholesky (sujeita a erros de arredondamento), é possível refinar essa solução da seguinte forma:

$$x = x^{(0)} + A^{-1}(b - A x^{(0)}) = x^{(0)} + e^{(0)} .$$

O vector $e^{(0)} = A^{-1}(b - A x^{(0)})$ é o vector erro já referido na secção 2.2.2. O residual pode ser calculado para a aproximação $x^{(0)}$ e é dado por $r^{(0)} = b - A x^{(0)}$.

Para calcular $x^{(1)}$ é necessário obter primeiro $e^{(0)}$ sem calcular a inversa de A. É possível realizar esta operação, resolvendo a equação

$$A e^{(0)} = r^{(0)}$$

Conjugando então as três equações

$$\begin{cases} r^{(0)} = b - Ax^{(0)} \\ Ae^{(0)} = r^{(0)} \\ x^{(1)} = x^{(0)} + e^{(0)} \end{cases}$$

podemos obter uma melhor aproximação $x^{(1)}$ à solução x. O sucesso deste procedimento, depende do cálculo do residual. Idealmente este valor deve tender para

zero, ou seja, a operação subtracção deve envolver quantidades muito próximas. Como já abordado na secção 2.2 esta operação deve ser realizada em precisão dupla para que a perda de significância nos dados seja mínima. Para repetição iterativa e condições de convergência consultar [Barret94, Kincaid02].

Os dois algoritmos, factorização *LU* e de Cholesky, têm sido objecto de estudo e têm sofrido alterações por parte da comunidade científica, no sentido de se adaptarem por um lado aos avanços tecnológicos ao nível das mais recentes arquitecturas dos computadores, e por outro às novas exigências ao nível das aplicações. Algumas versões das rotinas de álgebra linear BLAS e as bibliotecas de funções que constituem o LAPACK foram redesenhadas atendendo a estes pressupostos. Tópicos sobre este assunto, bem como a possibilidade de paralelizar partes do código no cálculo das matrizes *L* e *U*, encontram-se em [Kincaid02, Anderson95].

3.2 Métodos iterativos

Os métodos directos que resolvem o sistema de equações lineares do tipo (3.1), em precisão infinita produzem a solução exacta num número finito de passos. Quando a ordem do sistema aumenta (milhares de equações), o cálculo da solução por factorização ou usando outros métodos directos, como por exemplo a eliminação de Gauss, pode ser crítico em termos de tempo de execução num computador. Esta característica pode ser encarada como uma desvantagem já que parar a meio do processo implica não ter nenhuma aproximação à solução.

Os métodos iterativos ou indirectos, produzem uma sequência de vectores por passos que idealmente convergem para a solução. O processo pára ao fim de um número pré-estabelecido *M* de iterações ou quando se atinge uma precisão desejada. Note-se que, ao contrário dos métodos directos que utilizam toda a precisão quando funcionam, estes refinam a precisão à medida que convergem.

Em sistemas grandes, como os referidos no parágrafo anterior, se a precisão não for um requisito forte, em certos casos é possível aproximar uma solução do sistema em poucas iterações. Nomeadamente em sistemas esparsos, onde o número de zeros da matriz *A* é grande, os métodos iterativos revelam-se muito eficientes. Para além de existirem mecanismos óptimos para guardar em memória a matriz [Press92, Barret94], normalmente os métodos iterativos são estáveis e alguns erros, nomeadamente os de truncatura e arredondamento, podem ser amortecidos ao longo das iterações.

A resolução do sistema linear do tipo $Ax=b$ usando métodos iterativos é sugerida de forma natural quando se exprime a equação original sob a forma equivalente [Kincaid02]:

$$Qx = (Q - A)x + b \,. \tag{3.2}$$

Esta equação sugere um processo iterativo para o seu cálculo, ou seja,

$$Qx^{(k)} = (Q - A)x^{(k-1)} + b \quad , k \geq 1 \,.$$

O vector inicial $x^{(0)}$ pode ser arbitrário, se não existir uma melhor escolha: diz-se que o método converge, se a sequência $x^{(k)}$ convergir para a solução x do problema, para qualquer vector inicial $x^{(0)}$.

Em geral deve escolher-se a matriz de partição Q de forma a que:

- a sequência $x^{(k)}$ seja facilmente calculada;
- a sequência $x^{(k)}$ convirja rapidamente para a solução.

Estas duas condições ocorrem quando é fácil de resolver $Qx^{(k)}=y$ e Q^{-1} é uma boa aproximação a A^{-1} [Kincaid02].

3.2.1 Método de Jacobi

Este método deriva da análise isolada de cada uma das n equações do sistema $Ax=b$. Se para a i-ésima equação tivermos

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \,,$$

podemos resolver x_i assumindo que as outras entradas não variam

$$x_i = (b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j) \Big/ a_{ii} \,.$$

Esta equação sugere uma resolução iterativa da forma

$$x_i^{(k)} = (b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k-1)}) \Big/ a_{ii}$$

que normalmente se designa por iteração de Jacobi.

Neste caso a matriz Q da equação (3.2) é diagonal e os seus elementos são os mesmos da diagonal da matriz A do sistema. Um algoritmo para este método pode ser [Kincaid02]:

input n, (a_{ij}), (b_i) (x_i), M
for k= 1 **to** M **do**
  for i= 1 **to** n **do**

$$u_i \leftarrow \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j \right) \Big/ a_{ii}$$

  end do
  for i= 1 **to** n **do**

$$x_i \leftarrow u_i$$

  end do
  output k, (x_i)
end do.

Este algoritmo pode tornar-se mais eficiente se resolvermos todas as divisões antes de se entrar em ciclo. Assim teríamos,

…
for i= 1 **to** n **do**

$$d = 1/a_{ii}$$

$$b_i \leftarrow db_i$$

for j= 1 **to** n **do**

$$a_{ij} = da_{ij}$$

end do
end do
…

Nesta situação, o valor correspondente a u_i será

$$u_i \leftarrow b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j \ .$$

Se na matriz A do sistema todos os elementos da diagonal forem maiores do que zero, admitindo uma matriz diagonal $D=\boldsymbol{diag}(a_{ii})$, é possível realizar um pré-processamento ao sistema do tipo

$$D^{-1}Ax = D^{-1}b$$

de forma a que mais uma vez as divisões sejam evitadas [Kincaid02]. Temos então

$$\left(D^{-1/2}AD^{-1/2}\right)\left(D^{1/2}x\right)=\left(D^{-1/2}b\right)$$

onde a matriz $D^{1/2} = diag\left(\sqrt{a_{ii}}\right)$.

Em alguns sistemas iterativos, alguma atenção inicial à estrutura da matriz pode aumentar a velocidade ou convergência do método, contribuindo assim para uma melhor eficiência.

Muitas vezes, opta-se por este método quando se dispõem de máquinas paralelas, pois como a ordem das equações é irrelevante e as componentes para cada iteração de Jacobi são actualizadas simultaneamente, é possível realizar algumas destas tarefas ao mesmo tempo.

Apesar de intuitivo e de fácil implementação, normalmente a taxa de convergência é baixa.

3.2.2 Método de Gauss-Seidel

A possibilidade de utilizar na iteração k o valor residual de x_i logo que ele esteja disponível constitui a grande diferença em relação ao algoritmo de Jacobi.

Para obter o novo valor de x_i no método de Gauss-Seidel são precisos os valores actuais de $x_1, x_2 \ldots x_{i-1}$. Este facto faz com que o método de Gauss-Seidel tenha um fluxo computacional série, em contraste com o de Jacobi onde a actualização das componentes é feita em simultâneo.

Neste método a matriz Q da equação (3.2) é obtida extraindo a triangular inferior de A com os elementos da sua diagonal. Um algoritmo que realiza este método pode ser [Kincaid02]:

input n, (a_{ij}), (b_i) (x_i), M
for k= 1 **to** M **do**
  for i= 1 **to** n **do**

$$x_i \leftarrow \left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j \right) \Big/ a_{ii}$$

  end do
  output k, (x_i)
end do.

A ordem em Gauss-Seidel não é irrelevante e o valor de cada iteração depende da seriação das equações. Nomeadamente em matrizes esparsas, a ordem das equações pode afectar a taxa de convergência do método. Para mais detalhes pode consultar-se [Barret94].

O que foi dito no algoritmo anterior em relação ao aumento de eficiência devido ao pré-processamento da matriz continua a ser válido no método de Gauss-Seidel.

Apesar de existirem situações onde o Gauss-Seidel converge e o Jacobi diverge, em geral, se o Jacobi convergir, o Gauss-Seidel converge mais rápido, mas ainda com taxas pouco interessantes [Barret94].

3.2.3 Método *Sucessive overrelaxation*, *SOR*

Este método pode ser derivado do método de Gauss-Seidel através da introdução de um parâmetro w.

Uma iteração deste método pode representar-se por [Kincaid02]

$$x_i^{(k)} = w\breve{x}_i^{(k)} + (1-w)x_i^{k-1},$$

onde $\breve{x}_i^{(k)}$ representa o valor para a iteração de Gauss-Seidel. Se w =1 este algoritmo transforma-se no método de Gauss-Seidel como se pode comprovar no algoritmo a seguir apresentado.

input w, n, (a_{ij}), (b_i) (x_i), M
for k= *1* **to** M **do**
    for i= 1 **to** n **do**

$$u_i \leftarrow \left(b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j \right) \Big/ a_{ii}$$

    end do

$$x^{(k)} \leftarrow wu_i + (1-w)x^{(k-1)}$$

    output k, (x_i)
end do.

A representação matricial deste método pode descrever-se por,

$$x^{(k)} = (D - wL)^{-1}\left(wU + (1-w)D\right)x^{(k-1)} + w(D - wL)^{-1}b$$

onde D é uma matriz diagonal com os elementos da diagonal iguais aos da diagonal de A, $-L$ é a matriz triangular inferior de A, e –U é a respectivamente matriz triangular superior de A.

A escolha do parâmetro w depende entre outros factores da estrutura da matriz e deve ser escolhido de forma a acelerar a taxa de convergência para a solução: para existir convergência, o valor de w deve situar-se no intervalo [0, 2]. Algumas implementações mais sofisticadas estimam este parâmetro adaptativamente de forma a obter o valor de w que melhor taxa de convergência produza. Para mais detalhes ver [Golub89].

Para uma escolha óptima de w, o método *SOR* pode convergir uma ordem de grandeza mais rápido do que o de Gauss-Seidel [Barret94].

3.3 Métodos para matrizes especiais

3.3.1 Matrizes esparsas

Existem algoritmos específicos para a resolução de sistemas de equações lineares quando a estrutura da matriz é esparsa que tiram partido da forma como os elementos da matriz se encontram distribuídos [Bunch76]. As várias distribuições deste tipo de sistemas com a característica comum de conterem muitos zeros na matriz do sistema, estimulam métodos específicos a serem óptimos em relação ao tempo de resolução e ao espaço necessário em memória para guardar uma matriz deste tipo. As fórmulas de Sherman-Morrison e de Woodburry [Press92], contam-se entre alguns dos métodos que são usados para sistemas com este tipo de matrizes.

Para algumas situações a redução no número de operações pode passar de $\vartheta(n^3)$ para $\vartheta(n)$, nomeadamente em algoritmos para matrizes tridiagonais [Press92].

3.3.2 Matrizes Toeplitz

Da mesma forma que muitos métodos tentam reduzir sistemas genéricos a sistemas esparsos de resolução mais simples, é possível utilizar métodos que tirem partido do conhecimento acerca da forma como os elementos da matriz se encontram relacionados e distribuídos.

Um sistema de equações que envolva uma matriz A Toeplitz pode escrever-se como sendo

$$\sum_{j=1}^{n} a_{i-j} x_j = b_i \qquad , 1 \le i \le n.$$

Se a matriz for também simétrica, $[A]_{ij} = a_{|j-i|}$, pode utilizar-se o método de Levinson [Levinson47], que tem por princípio de funcionamento a utilização da recursividade na equação

$$\sum_{j=1}^{M} a_{i-j} x_j^{(M)} = b_i \ , \ \ 1 \le i \le M \, ,$$

onde $.^{(M)}$ representa o nível de recursividade. Quando $M=n$, atinge-se a precisão desejada.

De notar que à medida que se vai exigindo mais estrutura à matriz também o domínio das aplicações se torna mais específico. No entanto, este tipo de matrizes é muito frequente em problemas de predição linear, estimação espectral, projecto de filtros recursivos, códigos de correcção de erros e reconstrução de sinal, entre outros.

Existem algoritmos para resolver este tipo de sistemas propostos por vários autores, todos com a característica comum de terem complexidade algorítmica $\vartheta(n^2)$ [Durbin60, Berlekamp84, Zhang92]. Em [Vieira00] faz-se uma descrição e comparação entre os diferentes métodos para uma aplicação não linear de correcção de erros.

3.4 Métodos semi-iterativos

Este tipo de métodos é também muito usado na resolução de sistemas lineares quando as matrizes são esparsas não estruturadas e de tamanho médio. São métodos que com precisão infinita (exacta) têm um erro zero no máximo em n passos. Normalmente nestes métodos o número de iterações $M<<n$.

Quando a matriz é densa, o tempo gasto na sua factorização é comparável àquele necessário para resolver (semi-) iterativamente o problema. A vantagem neste caso vai para a factorização da matriz já que a possibilidade de calcular múltiplos valores para o vector b é evidente. Para além disto, se compararmos uma matriz densa com uma equivalente esparsa em relação à quantidade de memória ocupada, verificamos que uma

factorização triangular (quando possível) de uma matriz esparsa contém mais elementos diferentes de zero do que própria matriz.

Um conceito importante para a compreensão dos algoritmos que vamos analisar a seguir é a noção de forma quadrática associada a uma matriz. No capítulo anterior, na secção 2.1.1, definimos algumas propriedades das matrizes simétricas e positivas definidas. Agora podemos acrescentar que para matrizes deste tipo a forma quadrática $f(x)=x^TAx$ é minimizada pela solução do sistema de equações lineares $Ax=b$. Com este conceito presente, o entendimento acerca dos algoritmos que usam a variação da derivada em cada ponto (direcções conjugadas e gradiente) para pesquisa de valores óptimos torna-se mais claro. Para detalhes pode consultar-se [Shewchuk94].

LEMA 1, [Kincaid02]

- Forma quadrática associada a uma matriz -

Se uma matriz A for simétrica e positiva definida, o problema de resolver o sistema de equações $Ax=b$ é equivalente ao problema de minimizar a seguinte forma quadrática

$$f(x)=\langle x,Ax\rangle-2\langle x,b\rangle$$

■

Na figura 3.2, representa-se uma superfície quadrática bidimensional que pode servir de abstracção à pesquisa do valor óptimo para a solução do sistema quando a matriz do sistema é simétrica e positiva definida.

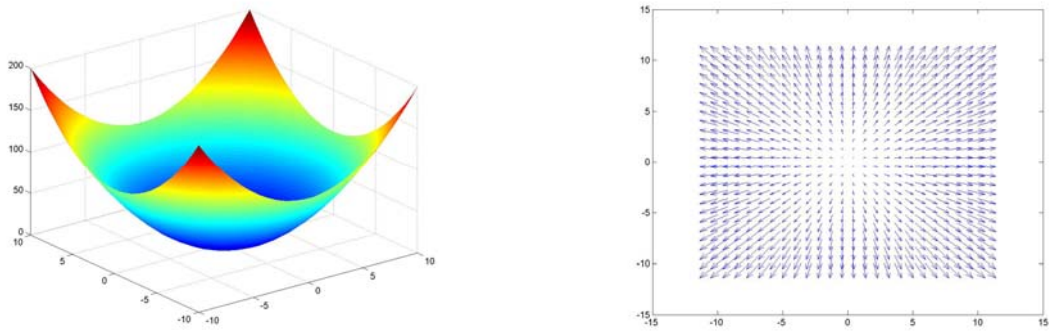

Figura 3.2: Parabolóide associado à forma quadrática com mínimo absoluto para o ponto (0, 0) e evolução do gradiente ao longo dessa superfície. O gradiente mínimo corresponde à solução do sistema de equações lineares $Ax=b$.

Nota-se claramente que no ponto óptimo (correspondente ao centro da figura do lado direito) o gradiente é mínimo. De referir ainda que para esta superfície quadrática existe apenas um mínimo global e não existem mínimos locais.

3.4.1 *Steepest Descent*

Utilizar este método para resolver o sistema de equações lineares do tipo 3.1 onde a matriz A de dimensão $n \times n$ é real, simétrica e positiva definida, consiste em produzir uma sequência de vectores a partir de um vector inicial x_0 que resultam da pesquisa do ponto mínimo para a superfície (figura 3.2) induzida pela forma quadrática associada à matriz A. A direcção que se escolhe em cada passo relaciona-se com aquela onde $f(x)$ mais decresce, ou seja, a direcção do residual r [Shewchuk94] definido na secção 2.2.2, ($v^{(k)}$ no pseudo código a seguir).

Uma descrição formal do método será,

input $x^{(0)}$, A, b, M
output 0, $x^{(0)}$
for k= 0 **to** M-1 **do**

$$v^{(k)} \leftarrow b - Ax^{(k)}$$

$$t_k \leftarrow \left\langle v^{(k)}, v^{(k)} \right\rangle \Big/ \left\langle v^{(k)}, Av^{(k)} \right\rangle$$

$$x^{(k+1)} \leftarrow x^{(k)} + t_k v^{(k)}$$

output k+1, $x^{(k+1)}$
end do.

Numa implementação prática deste algoritmo, os sucessivos vectores $x^{(0)}$, $x^{(1)}$,… não precisam de ser armazenados pois só nos interessa o que corresponde à iteração actual. O mesmo se pode afirmar para o vector das direcções $v^{(0)}$, $v^{(1)}$,…. Daqui resulta que o fluxo das descrições dos pseudo-códigos pretendem tornar inteligível o método e não serem computacionalmente eficientes.

A convergência deste método em função do número de condição da matriz é analisada em [Shewchuk94] onde se verifica que para $\kappa(A)$>40, a taxa de decréscimo no erro de iteração para iteração é pouco significativa, logo a convergência quando existe é lenta.

Quando falarmos de pré-condicionamento de matrizes na secção 3.5 voltamos a abordar a dependência da convergência em relação ao número de condição da matriz do sistema.

Na figura 3.3 podemos observar uma interpretação gráfica deste método.

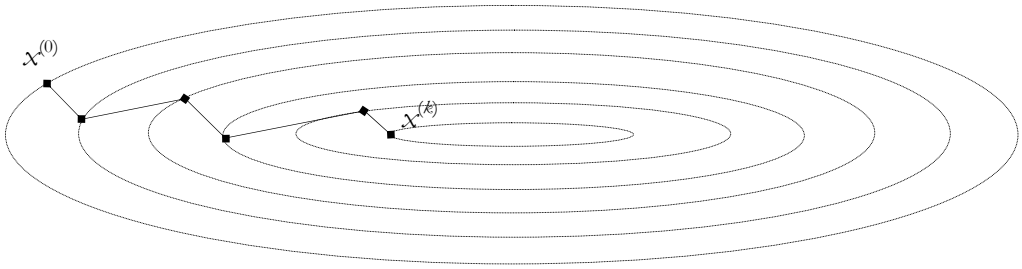

Figura 3.3: Interpretação gráfica do algoritmo *Steepest Descent* ao longo das curvas de nível da superfície quadrática.

Uma família de métodos, denominada de direcções conjugadas, partilha esta estratégia de minimizar a forma quadrática ao longo de sucessivas iterações. Estes algoritmos, para além de serem mais eficientes como vamos ver a seguir, diferem do descrito na figura 3.3, na forma como é tomada a direcção ao longo do processo iterativo.

3.4.2 Gradientes conjugados

A repetição de passos dados ao longo de direcções comuns no método anterior, sugeriram a Hestenes e Stiefel em 1952, uma outra abordagem, o método dos gradientes conjugados [Hestenes52].

Este, é um caso particular dos métodos das direcções conjugadas, e difere do *Steepest Descent* apenas no processo de encontrar a direcção que melhor aproxima a solução (minimiza a forma quadrática). Neste caso, as direcções são construídas por forma a que cada uma seja tomada apenas uma vez.

Quando o método foi inicialmente apresentado, criaram-se grandes expectativas já que os resultados teóricos admitiam que a solução era obtida em *n* passos para uma ordem *n*. Na prática, devido aos erros de arredondamento, e particularmente em problemas mal condicionados a aproximação à solução era pouco interessante. Para um método directo, pois era como tal que era considerado na altura, eram resultados pouco encorajadores.

Obter uma aproximação à solução ao fim de *n* passos, é um comportamento típico de um método iterativo. Na realidade, só passadas algumas décadas se percebeu esta ligação e o interesse por estes métodos ressurgiu novamente.

De facto, esperam-se respostas satisfatórias num número de passos inferior a *n* quando temos sistemas de ordem muito elevada. Para problemas bem condicionados, o número de iterações *M* necessárias para obter uma aproximação à solução satisfatória, pode ser muito mais pequeno do que a ordem do sistema [Sluis92].

TEOREMA 4, [Kincaid02]

- Sistema ortogonal -

Considere-se uma matriz A, simétrica e positiva definida de dimensão $n \times n$ e um conjunto de vectores diferentes de zero $\left\{v^{(1)}, v^{(2)}, \ldots, v^{(n)}\right\}$ que satisfazem à seguinte propriedade $\left\langle v^{(i)}, Av^{(j)} \right\rangle = \delta_{ij} \quad (1 \le i, j < n)$. Se definirmos um processo iterativo

$$x^{(i)} = x^{(i-1)} + \frac{\left\langle b - Ax^{(i-1)}, v(i) \right\rangle}{\left\langle v^{(i)}, Av(i) \right\rangle} v^{(i)} \qquad (1 \le i \le n)$$

onde $x^{(0)}$ é arbitrário, temos que $Ax^{(n)} = b$.

■

Uma descrição formal do algoritmo apresenta-se a seguir.

input $x^{(0)}$, A, b, M, ε

$r^{(0)} \leftarrow b - Ax^{(0)}$

$v^{(0)} \leftarrow r^{(0)}$

output 0, $x^{(0)}$, $r^{(0)}$
for k= 0 **to** M-1 **do**
  if $v^{(k)}$=0 **then** *exit loop*

  $t_k \leftarrow \left\langle r^{(k)}, r^{(k)} \right\rangle \Big/ \left\langle v^{(k)}, Av^{(k)} \right\rangle$

  $x^{(k+1)} \leftarrow x^{(k)} + t_k v^{(k)}$

  $r^{(k+1)} \leftarrow r^{(k)} - t_k Av^{(k)}$

  if $\left\| r^{(k+1)} \right\|_2^2 < \varepsilon$ **then** *exit loop*

    $s_k \leftarrow \left\langle r^{(k+1)}, r^{(k+1)} \right\rangle \Big/ \left\langle r^{(k)}, r^{(k)} \right\rangle$

    $v^{(k+1)} \leftarrow r^{(k+1)} + s_k v^{(k)}$

  output k+1, $x^{(k+1)}$, $r^{(k+1)}$
end do

Neste algoritmo, se $r^{(k)} = 0$ temos que $x^{(k)}$ é a solução (teórica) do sistema de equações lineares. A realização deste algoritmo necessita de armazenar quatro vectores $x^{(k)}$, $r^{(k)}$, $v^{(k)}$ e $Av^{(k)}$. Como já tínhamos referido este método não é recomendado para matrizes densas ou de banda. Por isso, apesar do produto $Av^{(k)}$ necessitar da matriz A, pode não ser necessário o armazenamento da matriz completa em memória, como se sugeriu na secção 3.3.1.

O esforço por iteração não é grande, e concentra-se essencialmente num produto de uma matriz por um vector $Av^{(k)}$, e no cálculo de dois produtos internos (depois da primeira iteração), $\left\langle v^{(k)}, Av^{(k)} \right\rangle$ e $\left\langle r^{(k+1)}, r^{(k+1)} \right\rangle$. De notar que a condição de paragem $\left\| r^{(k+1)} \right\|_2^2 = \left\langle r^{(k+1)}, r^{(k+1)} \right\rangle$ é obtida facilmente, pois envolve um produto interno entre uma quantidade calculada em cada iteração no passo anterior. Apresenta-se no apêndice A uma possível implementação deste método.

TEOREMA 5, [Kincaid02]

- Gradientes Conjugados -

Considere-se a matriz A, simétrica e positiva definida de dimensão $n \times n$. No algoritmo dos gradientes conjugados, para um inteiro qualquer $m<n$, se o conjunto dos vectores de direcção $\left\{p^{(1)}, p^{(2)}, \ldots, p^{(m)}\right\}$ for diferente de zero, temos que $r^{(i)} = b - A\,x^{(i)}$ para $0 \le i < m$, onde $\left\{r^{(1)}, r^{(2)}, \ldots, r^{(m)}\right\}$ é um conjunto de vectores residuais diferentes de zero e ortogonais, ou seja,

$$\left\langle r^{(i)}, r^{(j)} \right\rangle = 0 \quad \forall i \neq j \,.$$

■

Em [Shewchuk94] pode verificar-se que a convergência deste método é superior à do *Steepest Descent*, e mais uma vez se conclui que quanto pior for o condicionamento da matriz, $\kappa(A) \to \infty$, pior é a convergência do método.

Se a matriz for Toeplitz, a possibilidade de calcular Tx de forma rápida com a FFT em $\vartheta(n \log n)$ FLOPS pode tornar estes métodos muito favoráveis.

3.5 Pré-condicionamento de matrizes

Subjacente a este conceito está a ideia de 'melhorar' ou tornar a estrutura da matriz mais 'favorável'.

Algumas acções de pré-condicionamento tais como, equilíbrio de linha e equilíbrio de coluna [Kincaid02], ocorrem na factorização quando se usam métodos directos de resolução de equações lineares.

A taxa de convergência dos métodos iterativos depende fortemente das propriedades espectrais da matriz de iteração. Se for possível transformar o sistema de equações lineares dum dado problema num outro equivalente (com a mesma solução) mas com uma matriz associada de estrutura mais favorável, realiza-se o conceito de pré-condicionamento de uma matriz. A matriz que efectua esta transformação chama-se matriz de pré-condicionamento.

Admitindo que M é uma matriz simétrica, positiva definida que aproxima A mas é mais simples de inverter, pode resolver-se $Ax=b$ indirectamente a partir do sistema

$$M^{-1}Ax=M^{-1}b.$$

Se $\kappa(M^{-1}A)<<\kappa(A)$ ou o espectro da matriz $\sigma(M^{-1}A)$ estiver mais concentrado, pode resolver-se o sistema equivalente agora de uma forma mais eficiente.

Muitos métodos iterativos dependem em grande parte do pré-condicionamento da matriz do problema. Existem mesmo situações para as quais só com a aplicação da matriz de pré-condicionamento é possível existir convergência. Apesar deste facto, um compromisso importante deve ser tomado entre o custo de encontrar e aplicar a matriz de pré-condicionamento e o ganho que se obtém na velocidade de convergência do sistema. Uma situação favorável será o caso quando a matriz M for circulante, já que a sua inversa pode ser obtida facilmente com um esforço computacional da ordem $\vartheta(n\log n)$, a partir dos seus valores próprios e da matriz unitária de Fourier como se explica na secção 2.1.2.

3.5.1 Gradientes conjugados com pré-condicionamento

Trata-se de um método para a resolução do sistema $Ax=b$, onde a matriz do sistema é simétrica e positiva definida. Considerando uma matriz S não singular, um novo sistema pode escrever-se

$$\widetilde{A}\widetilde{x}=\widetilde{b}$$

onde,

$$\begin{cases}\widetilde{A}=S^{T}AS\\ \widetilde{x}=S^{-1}x\\ \widetilde{b}=S^{T}b\end{cases},$$

e a relação entre os números de condição deve obedecer a $\kappa(\widetilde{A})<<\kappa(A)$. Como será desejável, o método iterativo utilizado para resolver o sistema pré-condicionado deve convergir mais rápido do que o original.

Vamos admitir a matriz Q simétrica e positiva definida que pode ser factorizada a partir de S da seguinte forma:

$$Q^{-1}=S\,S^{T}.$$

Partindo do pseudo código dos gradientes conjugados, poderíamos escrever para o método dos gradientes conjugados com pré-condicionamento:

$\widetilde{r}^{(0)} \leftarrow \widetilde{b} - \widetilde{A}\widetilde{x}^{(0)}$

$\widetilde{v}^{(0)} \leftarrow \widetilde{r}^{(0)}$

for k= 0 **to** M-1 **do**

$\widetilde{t}_k \leftarrow \left\langle \widetilde{r}^{(k)}, \widetilde{r}^{(k)} \right\rangle \Big/ \left\langle \widetilde{v}^{(k)}, \widetilde{A}\widetilde{v}^{(k)} \right\rangle$

$\widetilde{x}^{(k+1)} \leftarrow \widetilde{x}^{(k)} + \widetilde{t}_k \widetilde{v}^{(k)}$

$\widetilde{r}^{(k+1)} \leftarrow \widetilde{r}^{(k)} - \widetilde{t}_k \widetilde{A}\widetilde{v}^{(k)}$

$\widetilde{s}_k \leftarrow \left\langle \widetilde{r}^{(k+1)}, \widetilde{r}^{(k+1)} \right\rangle \Big/ \left\langle \widetilde{r}^{(k)}, \widetilde{r}^{(k)} \right\rangle$

$\widetilde{v}^{(k+1)} \leftarrow \widetilde{r}^{(k+1)} + \widetilde{s}_k \widetilde{v}^{(k)}$

end do.

Apesar de ser possível implementar este algoritmo, a estrutura esparsa inicial de A pode ser completamente alterada quando se constrói $\widetilde{A}$. É possível uma descrição alternativa utilizando as seguintes relações e equivalências:

$\widetilde{x}^{(k)} = S^{-1}x^{(k)}$

$\widetilde{v}^{(k)} = S^{-1}v^{(k)}$

$\widetilde{r}^{(k)} = \widetilde{b} - \widetilde{A}\widetilde{x}^{(k)} = S^{T}b - (S^{T}AS)(S^{-1}x^{(k)}) = S^{T}r^{(k)}$.

$\bar{r}^{(k)} = Q^{-1}r^{(k)}$

Em seguida apresenta-se o pseudo código para o método dos gradientes conjugados com pré-condicionamento onde se usa o sistema original e o pré-condicionamento é feito implicitamente ao longo do algoritmo [Kincaid02].

input $x^{(0)}$, A, b, M, Q

$r^{(0)} \leftarrow b - Ax^{(0)}$

solve $Q\bar{r}^{(0)} = r^{(0)}$ for $\bar{r}^{(0)}$

$v^{(0)} \leftarrow r^{(0)}$

output 0, $x^{(0)}$

for k= 0 **to** M-1 **do**

if $v^{(k)}$=0 **then** *exit loop*

$\tilde{t}_k \leftarrow \left\langle \bar{r}^{(k)}, r^{(k)} \right\rangle \Big/ \left\langle v^{(k)}, Av^{(k)} \right\rangle$

$x^{(k+1)} \leftarrow x^{(k)} + \tilde{t}_k v^{(k)}$

$r^{(k+1)} \leftarrow r^{(k)} - \tilde{t}_k Av^{(k)}$

solve $Q\bar{r}^{(k+1)} = r^{(k+1)}$ for $\bar{r}^{(k+1)}$

if $\left\langle \bar{r}^{(k+1)}, r^{(k+1)} \right\rangle < \varepsilon$ **then**

if $\left\langle r^{(k+1)}, r^{(k+1)} \right\rangle < \varepsilon$ **then** *exit loop*

end if

$\tilde{s}_k \leftarrow \left\langle \bar{r}^{(k+1)}, r^{(k+1)} \right\rangle \Big/ \left\langle \bar{r}^{(k)}, r^{(k)} \right\rangle$

$v^{(k+1)} \leftarrow \bar{r}^{(k+1)} + \tilde{s}_k v^{(k)}$

output k+1, $x^{(k+1)}$, $r^{(k+1)}$

end do.

Este algoritmo reduz-se ao do gradientes conjugados quando Q^{-1}=I, já que para esta situação $\bar{r}^{(k)} = r^{(k)}$, $\tilde{t}_k = t_k$, e $\tilde{s}_k = s_k$.

Se Q=A e $S = A^{-1/2}$, o sistema pré-condicionador reduz-se a $\tilde{x}^{(k)} = \tilde{b}$ e é de resolução trivial. Infelizmente, esta condição ideal ($\kappa(\tilde{A})$=1) em termos computacionais

não é interessante, pois determinar o valor de $\widetilde{b} = S^T b$ é tão difícil como resolver o sistema original.

Como tem de resolver-se em cada iteração um sistema da forma $Qx=y$, a matriz Q deve ser escolhida de tal forma que o sistema seja fácil de resolver. Se Q for uma matriz diagonal esta condição está garantida, mas se pretendermos velocidade na convergência teremos que escolher matrizes de pré-condicionamento mais complicadas. À medida que Q^{-1} se torna uma melhor aproximação de A, o novo sistema fica melhor condicionado e a convergência do processo iterativo ocorre rapidamente em poucos passos. Por outro lado, a resolução do sistema $Qx=y$ torna-se mais difícil. Daqui fica claro que um compromisso importante deve ser feito na escolha entre processos iterativos com poucos passos mas computacionalmente dispendiosos, e aqueles que necessitam de muitos passos mas menos dispendiosos. Apresenta-se no apêndice A uma possível implementação para este método.

Algumas bibliotecas de funções disponíveis em muitos computadores incluem hoje em dia já algumas das versões mais evoluídas dos algoritmos para a resolução de equações lineares que atrás se apresentaram, incluindo o pré-condicionamento de matrizes para o método dos gradientes conjugados. Alguns exemplos destas bibliotecas são: ITPACKV, NSPCG, PCGPAK2 e PCG que se apresentam em [Kincaid02].

CAPÍTULO 4

Dos códigos convencionais aos códigos DFT

Um dos objectivos deste capítulo é enquadrar historicamente a TCCE estabelecendo a sua íntima relação com a Teoria da Informação. Deste modo, pretendemos dar uma perspectiva relacionada com o aparecimento dos primeiros códigos de correcção de erros que eram sobre corpos finitos, e da forma como podemos relacionar essas mesmas questões com os que agora nos propomos apresentar, isto é, códigos sobre o corpo dos reais/complexos baseados na DFT (códigos DFT).

Nos primeiros pontos, discutem-se questões relativas à Teoria da Informação, enquanto que nos seguintes se abordam assuntos sobre códigos de correcção de erros.

As relações existentes entre os códigos sobre corpos finitos, a amostragem, e os códigos DFT são apresentadas no fim e servem de referência para o capítulo seguinte que se considera o mais importante, já que nele se sugere a resolução de problemas de correcção de erros baseada em códigos DFT de dimensão mínima.

4.1 Teoria da Informação

4.1.1 Entropia

A Teoria da Informação responde a duas questões fundamentais nas comunicações: **"Qual a taxa máxima de compressão?"** e **"Qual a taxa de transmissão máxima**

numa comunicação?". Os conceitos fundamentais que permitem responder a estas perguntas são a **entropia** e a **capacidade de canal** [Shannon48, Shannon49].

Para definir entropia, uma ideia fundamental deve estar sempre presente: "**A informação adquirida ao saber que ocorreu um determinado evento é tanto maior quanto menor for a probabilidade de ocorrência desse acontecimento**".

A função logaritmo pode ser usada para definir a **informação** I_s, associada a um acontecimento s com probabilidade p

$$I_S = \log_2 {}^{1}\!/_{p}\,.$$

O aumento de informação é inversamente proporcional à probabilidade de ocorrência do evento o que justifica a fracção ${}^{1}\!/_{p}$. A função logarítmica, força o valor zero quando se sabe à partida que o acontecimento tem probabilidade p=1, já que neste caso a informação é nula. Para além destes factos, este resultado é congruente com a probabilidade do produto de dois eventos independentes,

$$\log_2 {}^{1}\!/_{p(a.b)} = -\left(\log_2 p_a + \log_2 p_b\right) = I_a + I_b\,.$$

O valor médio desta grandeza é a entropia.

DEFINIÇÃO 3

- Entropia-

A entropia H de uma fonte que produz n símbolos a partir de um alfabeto $s_1, s_2,\ldots,s_n$, sendo a probabilidade do símbolo s_i igual a p_i é

$$H(p_1, p_2, \ldots p_n) = -\sum_{k=1}^{n} p_k \log_2 p_k\,, \quad \text{(bits/símbolo)}.$$

■

Usa-se a convenção $0 \log_2 0 = 0$ já que $x \log_2 x \to 0$ quando $x \to 0$, e a adição de termos com probabilidade nula não contribui para o aumento da incerteza.

Assumindo como aqui que o logaritmo tem base 2, a entropia é medida em bits. Sendo assim, representa o número médio de bits necessários para descrever uma fonte.

O valor da entropia é máximo quando $p_i=1/n$ e quando isto acontece tem-se

$$H(p_1, p_2, \ldots p_n) = -\sum_{k=1}^{n} \frac{1}{n} \log_2 \frac{1}{n} = (n-1+1)\frac{1}{n}\log_2\left(\frac{1}{n}\right)^{-1} = \log_2 n .$$

■

Admita-se uma fonte binária com um alfabeto de dois símbolos de probabilidades p e $1-p$

$$X = \begin{cases} 1 & \text{, com probabilidade } p \\ 0 & \text{, com probabilidade } 1 - p \end{cases}.$$

A entropia $H(X)$ é dada pela expressão

$$H(X) = -p\log_2 p - (1-p)\log_2(1-p),$$

e na figura 4.1 temos a sua representação gráfica.

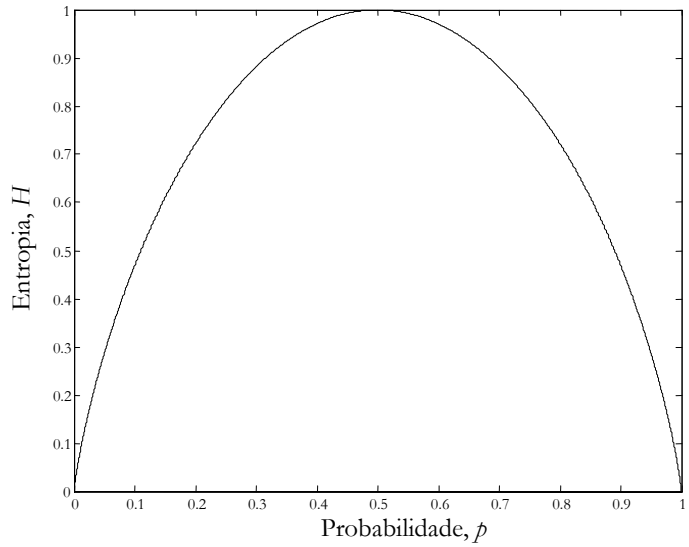

Figura 4.1: Entropia versus probabilidade.

Quando a probabilidade dos dois símbolos é igual, p=0.5, a função entropia tem o valor mais elevado. A incerteza é máxima já que a probabilidade de ocorrer qualquer um dos símbolos é igual.

Quando se sabe à partida que uma das palavras nunca ocorre, p=1 ou p=0, a entropia é zero. A quantidade de informação é naturalmente zero, pois o acontecimento passa a ser completamente previsível.

4.1.2 Capacidade de um canal

Outro resultado não menos importante de Shannon estabelece um limite máximo para a taxa de transmissão numa comunicação com probabilidade de erro nos dados fixa, mas arbitrariamente baixa.

Pode definir-se comunicação entre dois pontos A e B como sendo o fenómeno físico em A que induz alterações no estado físico de B. Esta transferência de informação é sujeita a imperfeições com características e origens diferentes. A comunicação será bem sucedida quando receptor e emissor concordarem quanto à informação enviada.

Quando se usa o canal de comunicação n vezes, o número máximo de símbolos que se distinguem no receptor cresce exponencialmente, sendo o valor máximo definido como a capacidade do canal. A analogia matemática do processo físico de uma comunicação pode ser descrita pelo diagrama de blocos da figura 4.2.

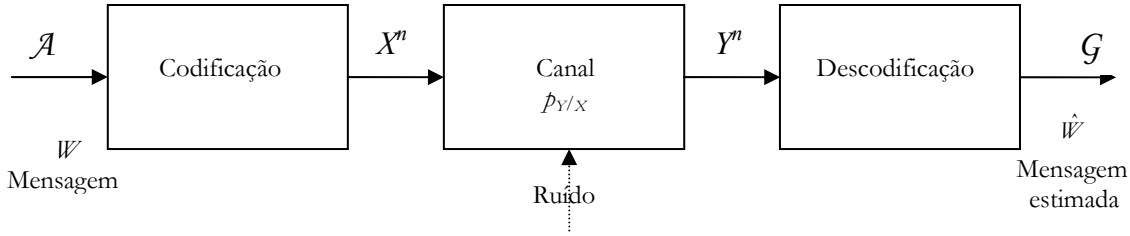

Figura 4.2: Canal de Comunicação.

As palavras na fonte são geradas a partir de um alfabeto finito $\mathcal{A}$ e um sub-conjunto de $\mathcal{A}$ forma uma sequência de símbolos de canal X^n. As características e propriedades do canal personalizam os dados e é produzida a sequência Y^n. Apesar da sequência de saída ser aleatória a sua distribuição depende da sequência de entrada. É a partir desta que se tenta reconstruir a mensagem original.

DEFINIÇÃO 4

– Canal discreto –

Define-se um canal discreto como sendo um sistema com um alfabeto de entrada $\mathcal{A}$, um alfabeto de saída $\mathcal{G}$ e uma matriz de probabilidade de transição $p_{Y/X}$ que exprime a probabilidade de observarmos o símbolo Y quando é enviado o símbolo X. Um canal sem memória é aquele onde a probabilidade da distribuição da saída depende apenas da entrada naquele instante e é independente de outras quaisquer entradas ou saídas.

■

Por analogia com a definição 3 podemos definir entropia condicional como um caso particular da entropia relativa, que representa uma medida da distância entre duas distribuições de probabilidades.

DEFINIÇÃO 5 [Cover91]

– Entropia condicional –

A entropia de uma variável aleatória Y dada uma outra variável aleatória X é uma medida da quantidade de informação que uma tem acerca da outra. Designamo-la por **informação mútua** ou **entropia condicional**, tendo-se:

$$I(X;Y) = H(X) - H(X \mid Y) = \sum_{x,y} p_{x,y} \log_2 \frac{p_{x,y}}{p_x p_y}$$

■

Duas sequências diferentes de entrada não devem provocar a mesma saída, sob pena da recuperação se tornar impossível. Uma das soluções é escolher apenas um sub-conjunto de sequências de entrada, de tal forma que com probabilidade elevada se consiga saber qual a palavra de entrada que induz cada símbolo na saída. Assim, consegue-se reconstruir a mensagem original a partir do receptor com uma probabilidade de erro fixa, mas suficientemente baixa para poder ser desprezada. **A taxa máxima com que se consegue realizar esta tarefa define-se como capacidade do canal**.

Podemos definir capacidade de um canal em função da informação mútua.

DEFINIÇÃO 6

- Capacidade de um canal -

A capacidade de um canal sem memória é

$$C = \max_{p_x} I(X;Y),$$

onde o máximo é calculado no conjunto de todas as possíveis distribuições de entrada p_x.

■

O teorema da codificação de canal de Shannon prova que para qualquer probabilidade de erro dada ε, existe pelo menos um código que possibilita a transmissão com uma probabilidade de erro inferior ou no limite igual a esse valor, abaixo da capacidade. Três ideias fundamentais contribuíram para a existência deste teorema: admitir uma probabilidade de erro pequena mas diferente de zero; admitir que a utilização do canal possa ser realizada tantas vezes quanto as necessárias; considerar o valor médio das probabilidades de erro sobre um conjunto aleatório de códigos.

TEOREMA 6, [Cover91]

- Teorema da codificação de canal -

Para uma dada taxa de transmissão $R < C$ existe uma sequência de códigos com uma probabilidade máxima de erro convergente para zero. Reciprocamente, qualquer sequência de códigos com probabilidade de erro convergente para zero implica $R \leq C$.

■

Obter probabilidades de erro baixas com códigos simples, ou seja, com mecanismos eficientes de codificação e descodificação, passou a ser desde então o desiderato principal dos investigadores na nova área do saber então criada. O desafio à engenharia da altura, de transmitir informação sem erros tentando construir canais perfeitos, passou também a ser o de encontrar o método ideal para realizar tal tarefa.

4.2 Códigos de correcção de erros

A partir de 1950 realizaram-se esforços para encontrar classes de códigos que produzissem as prometidas probabilidades de erros arbitrariamente baixas.

Duas abordagens diferentes surgiram conduzindo aos códigos por bloco e aos códigos convolucionais.

4.2.1 Códigos por bloco e convolucionais

Os primeiros códigos construídos no início da década de 1950 devem-se a Hamming [Hamming50]. Só passados 10 anos surgiram avanços substanciais quando Bose e Ray-Chaudhuri (1960) e Hocquenghem (1959) encontraram um conjunto mais alargado de códigos de correcção múltipla de erros que passaram a ser conhecidos por códigos

BCH. Também nessa altura Reed e Solomon (1960) apresentaram uma classe de códigos para o caso em que os canais não são binários. A descoberta dos códigos BCH traduziu-se num salto qualitativo importante na altura, mas a implementação prática do codificador/descodificador era um desafio à tecnologia vigente. Berlekamp e Massey descobriram um meio de realizar essa tarefa e o advento dos circuitos integrados tornou realizável em hardware algoritmos com níveis de complexidade elevados, mas com melhores performances. Por exemplo o disco compacto, vulgo CD, inclui um circuito de correcção baseado em dois códigos de Reed-Solomon entrelaçados que permitem ao descodificador corrigir até 4000 erros.

Todos os códigos por bloco têm uma característica em comum: cada símbolo da saída depende apenas de um símbolo da entrada. A outra abordagem referida no início designa-se por códigos convolucionais e são algoritmos que fazem descodificação sequencial e onde o valor de cada saída depende também de entradas anteriores. O seu comprimento é indefinido ao contrário dos códigos de bloco e normalmente são representados em árvore, tendo o descodificador a tarefa de pesquisar essa árvore para encontrar os símbolos de entrada. Em 1967, surgiu o algoritmo de Viterbi que ganhou popularidade ao ser utilizado para descodificar códigos de modesta complexidade. A teoria destes códigos desenvolveu-se consideravelmente nos últimos 25 anos e pode ser analisada nos textos [Heegard99, Vucetic00, Blahut03].

O interesse por este assunto aumentou, estimulado pela evolução tecnológica que contribuiu em grande parte para este incremento. Novos códigos apareceram nas décadas seguintes contribuindo para os aspectos teóricos se tornarem mais robustos e explícitos de tal forma que hoje em dia a maior parte dos sistemas de comunicação ou de armazenamento digitais têm um bloco de codificação/descodificação.

Apesar duma associação natural às comunicações, onde a existência de limitações na potência a transmitir pode ser compensada pela introdução de um código de modo a que as mensagens de fraca potência no receptor sejam recuperadas sem erros, existem outras importantes aplicações onde estes códigos estão presentes, tais como: as técnicas de compressão de dados, a protecção de dados em memória ou discos de computadores ou a protecção contra mal-funcionamento ou ruído em circuitos lógicos.

Na década de 90 uma nova classe de métodos com aspectos inovadores despertaram muito interesse. Os códigos que usam a concatenação paralela com permutadores, conhecidos por Turbo-Códigos, descobertos por Berrou, Glavieux e Thitimajshima em

1993 [Berrou93, Berrou96], são da maior importância nas telecomunicações e despertaram novo interesse nos investigadores desta área permitindo atingir a menos de fracções de 1 dB o limite de Shannon para probabilidades de erro na ordem de 10^{-4} ou 10^{-5} [Benedetto96, Hagenauer96, Perez96].

4.2.1.1 Conceitos elementares relativos a códigos

Um código binário de comprimento M e tamanho do bloco N é um conjunto de M símbolos (ou palavras) binários com N bits. Genericamente, $M=2^K$ com $K\in\mathbb{N}$, e designa-se por código binário (N, K).

Admitindo $M=4$ e $N=5$, a título de exemplo, podemos utilizar o código definido pela matriz

$$C = \begin{bmatrix} 1\ 0\ 1\ 0\ 1 \\ 1\ 0\ 0\ 1\ 0 \\ 0\ 1\ 1\ 1\ 0 \\ 1\ 1\ 1\ 1\ 1 \end{bmatrix}, \tag{4.1}$$

para representar todos os números binários com dois bits. Mais precisamente, tem-se a correspondência

$$\begin{aligned} 0\ 0 &\leftrightarrow 1\ 0\ 1\ 0\ 1 \\ 0\ 1 &\leftrightarrow 1\ 0\ 0\ 1\ 0 \\ 1\ 0 &\leftrightarrow 0\ 1\ 1\ 1\ 0 \\ 1\ 1 &\leftrightarrow 1\ 1\ 1\ 1\ 1 \end{aligned}.$$

Se não ocorrerem erros, a entrada é correctamente descodificada. A ocorrência de um erro pode ser resolvida fazendo com que por comparação, se estime com o símbolo mais parecido. Ou seja, se a palavra errada for '01100' admite-se que a entrada original seria '10'.

O facto de certos canais não serem binários, faz com que, por razões de performance os códigos e os alfabetos também não o sejam. Em geral os códigos por blocos formulam-se em corpos finitos, e usam um alfabeto finito de p símbolos, $\{0,1,2,\ldots,p-1\}$. Quando $p=2$, os símbolos designam-se bits. Os códigos são implementados em corpos finitos de Galois GF(q) [Brison97, Rotman98].

DEFINIÇÃO 7

- Código por blocos -

Um código por blocos em que o sinal mensagem (ou vector) a codificar tem K elementos e o símbolo codificado tem uma dimensão $N > K$ é designado por código do tipo (N, K).

■

A taxa do código R pode definir-se como,

$$R = \frac{K}{N}.$$

DEFINIÇÃO 8

- Código por blocos sistemáticos -

Um código por blocos do tipo (N, K) é sistemático quando o símbolo codificado é constituído pela palavra original K mais N-K bits normalmente designados de paridade.

■

Na figura 4.3 apresenta-se de uma forma simplificada, um diagrama de blocos que traduz um código por blocos sistemático.

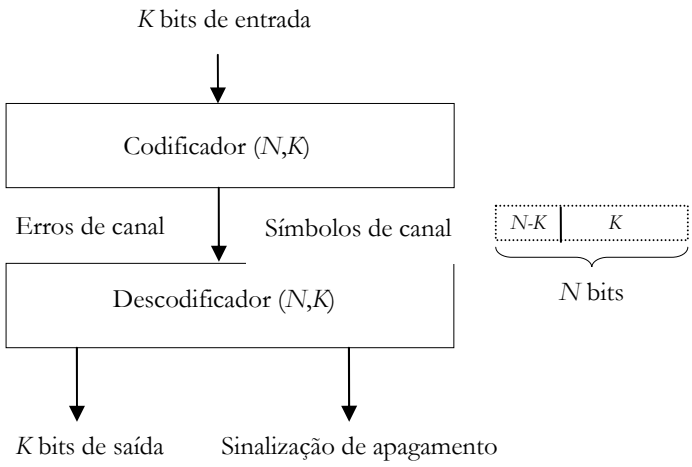

Figura 4.3: Fluxo de um código por blocos.

A sinalização de apagamento, quando ocorre, indica a posição onde os bits foram corrompidos.

Os códigos por blocos são caracterizados pelo tamanho do bloco N, pela quantidade de informação K, e pela sua distância mínima d^r, definida em termos da distância de Hamming:

DEFINIÇÃO 9

- Distância de Hamming -

A distância de Hamming $d(a, b)$ entre dois símbolos a e b de um código é dada pelo número de bits em que a e b diferem.

■

Podemos agora definir distância mínima.

DEFINIÇÃO 10

- Distância mínima de um código -

A distância mínima de um código é a distância de Hamming do par de símbolos do código onde essa distância é menor. Um código por blocos (N, K) com uma distância mínima d^r representa-se por (N, K, d^r).

■

Genericamente, se ocorrerem t erros e se a distância entre o símbolo recebido e todos os outros que pertencem ao código for maior do que t, o descodificador pode corrigir o erro assumindo que o símbolo mais próximo do recebido foi aquele que se enviou.

TEOREMA 7, [Blahut83]

- Limite de Singleton -

A distância de Hamming mínima entre dois símbolos de qualquer código linear satisfaz

$$d \leq 1+N-K.$$

■

Por análise deste teorema pode concluir-se que para um código conseguir corrigir t erros é necessário que este tenha pelo menos $2t$ bits de paridade. Quando o número de

bits de paridade for igual a $2t$ a distância mínima é igual a $1+N-K$, e o código é designado por Código de Distância Máxima (CDM) [Blahut83].

DEFINIÇÃO 11

- Peso de um código -

O peso de um código linear é igual ao valor mínimo da distância de Hamming de qualquer símbolo desse código ao vector nulo.

■

4.3 Codificação de fonte e de canal

O bloco de codificação apresentado na figura 4.2 pode ser decomposto em três distintos: um de codificação de canal, outro de fonte e um final de modulação, figura 4.4. Por analogia, no receptor são realizadas as operações de descodificação por ordem inversa. Neste trabalho não se discutem questões relativas aos blocos de modulação e desmodulação.

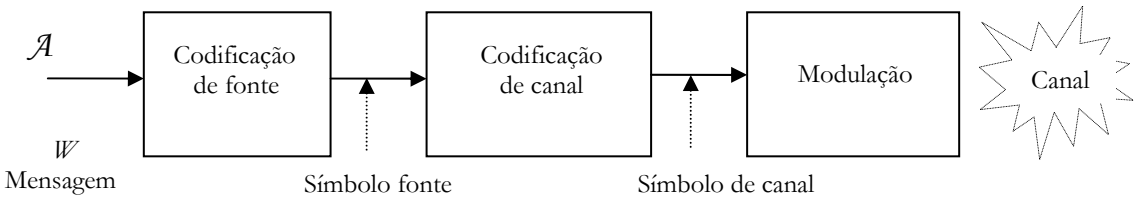

Figura 4.4: Bloco de codificação.

Numa transmissão, a necessidade de codificar informação de maneiras distintas, relaciona-se em grande parte com as características dos dados e as do canal. Ou seja, a codificação de fonte tenta representar o sinal de entrada com o menor número de bits possível, enquanto que se pretendermos proteger os dados de ruído do meio de transmissão, necessitamos de fazer codificação de canal.

Pode assim classificar-se a codificação em dois grandes tipos: de fonte e de canal. A codificação de fonte retira redundância do sinal que se pretende transmitir, diminuindo assim a sua largura de banda (LB). Por sua vez, a codificação de canal protege os dados do meio fazendo com que o número de bits a serem transmitidos e a LB aumente. Os trabalhos de [Benedetto99, Blahut83] abordam estes dois tipos de codificações.

Se porventura a codificação de fonte não for realizada, são enviados mais bits do que os necessários para o sinal ser transportado para o receptor. Quando a codificação de canal não existe, é transmitida apenas informação directamente proveniente da fonte mas com uma taxa de erro nos bits mais elevada.

Apesar do contra senso aparente, pois por um lado o sinal é comprimido e por outro acrescentam-se mais bits, este procedimento justifica o facto da codificação ser estratégica e responder a objectivos distintos. De referir que alguns trabalhos sugerem também técnicas de codificação canal/fonte em conjunto [Proakis95, Azami96, Frias97]. No capítulo seguinte voltamos a este assunto no contexto de códigos de correcção de erros em corpos reais.

4.4 Erros e apagamentos em canais de comunicação

Ao transmitir um sinal *f* através dum canal de comunicação como o descrito na secção 4.1.2, verifica-se normalmente que o sinal observado *g* do lado do receptor é uma versão modificada de *f*. Se considerarmos que o ruído indesejável que levou à alteração de *f* é adicionado ao sinal original, podemos estabelecer a seguinte relação

$$g = f + r, \tag{4.2}$$

onde *r* representa o ruído ou sinal de erro (neste caso aditivo).

A caracterização dos vários tipos de ruído [Ott76, Schwartz90, Looney97], bem como o estudo de técnicas para a sua eliminação ou atenuação, constituem um dos problemas mais importantes e mais estudados em diversas áreas da engenharia. No que nos diz respeito, podemos referir as telecomunicações e a utilização do processamento de sinal (PDS) para resolver problemas associados a erros que ocorrem em canais de comunicação. O ruído pode manifestar-se de diversas maneiras. Pode sentir-se em todo o conjunto de dados como acontece no processo de quantificação numa ADC ou apenas num subconjunto quando existe um conjunto incompleto de dados. O esforço empreendido nesta área tornou possível lidarmos com determinadas formas de ruído como é o caso do ruído aditivo Gaussiano, mas ainda não é suficiente para compreender ou prever eficazmente por exemplo o ruído impulsivo.

No contexto de sinais digitais, é possível relacionar o ruído impulsivo com erros, ou seja, se pensarmos no envio de *f* através de um canal de comunicação, podemos admitir que o subconjunto de amostras alteradas ou corrompidas do sinal observado *g* se deve

ao ruído impulsivo. Em muitos casos, à impossibilidade de distinguir as amostras correctas das incorrectas (posição dos impulsos ou localização dos erros), associa-se o facto do número de erros ser também desconhecido, o que complica ainda mais o problema.

A adição de redundância controlada aos dados no emissor e a sua validação no receptor é um dos processos que existem para se obter um conhecimento parcial ou total da localização dos erros. Um outro exemplo consiste na técnica usada pelos sistemas de transmissão com comutação de pacotes, tais como os IP ou ATM [Stallings88, Black95].

Estes erros que ocorrem em posições conhecidas designam-se por **apagamentos** e um canal sujeito a este tipo de erros designa-se por **canal com apagamentos**. Neste tipo de canais, quando ocorrem erros é apenas necessário corrigir a amplitude das amostras alteradas, já que as suas localizações são conhecidas a priori.

Quando não se conhecem nem a posição nem o número de erros, tipicamente, a estratégia associada à **correcção de erros** pode ser decomposta em duas tarefas pela seguinte ordem:

- cálculo da localização dos erros; fase que vamos chamar por **detecção de erros** que é feita com recurso a técnicas não lineares;
- conhecidas as posições e o número de erros, devem corrigir-se as suas amplitudes; tarefa resolvida com técnicas lineares que vamos designar por **correcção de apagamentos**.

Daqui para a frente quando falarmos em correcção de erros estamos a considerar a detecção dos erros e a correcção de amplitudes das amostras corrompidas. A designação correcção de apagamentos relaciona-se apenas com a correcção da amplitude dos erros. De notar que, para além do número de quantidades desconhecidas no caso da correcção de apagamentos ser sensivelmente metade quando comparado com os da correcção de erros, a necessidade de utilização de técnicas não lineares por parte destes torna o problema ainda mais complicado. No capítulo seguinte são discutidos algoritmos para correcção de erros e apagamentos.

A maior ou menor dificuldade de corrigir erros ou apagamentos depende em grande parte do padrão dos erros ao longo do tempo. Ou seja, se estes ocorrem durante um dado intervalo contínuo (**tipo rajada)**, a correcção é mais complicada quando comparada com a situação em que eles ocorrem distribuídos no tempo (**tipo esparso)**.

Os erros do tipo rajada são muito comuns em transmissão multimédia e podem ser originados, por:

- perda de pacotes em redes de comutação de pacotes;
- atraso excessivo em aplicações de tempo real;
- falhas em sistemas de armazenamento (CD, DVD, suporte magnético).

O entrelaçamento das amostras conta-se entre as técnicas mais simples e mais usadas no sentido de espalhar as perdas contíguas de amostras [Berman94, Yu96, Soares98]. Na realidade, só quando a probabilidade de erros do tipo rajada for elevada se deve optar pelo entrelaçamento das amostras já que a sua utilização sem regra pode fazer com que padrões de erros aleatórios à partida se transformem em contínuos.

4.5 Processamento de sinal, amostragem e códigos reais

A estimação do sinal f dado pela expressão (4.2) a partir da equação $g=f+r$ sob condições ideais onde se admite o espectro do ruído ortogonal ao do sinal f, é trivial usando filtragem linear. Na prática esta situação é muito imprevisível, e na grande maioria dos casos (senão em todos), o ruído tem também uma componente do espectro com frequências comuns às do sinal f. A utilização dum procedimento equivalente à situação ideal conduziria a fenómenos de distorção no sinal estimado para f mais ou menos graves, função das características do ruído e dos filtros.

Quando existe sobre-amostragem, uma estratégia muito comum de estimar f a partir do sinal observado $g=f+r$ consiste em filtrar a componente do espectro de r que não se sobrepõe ao espectro de f. No entanto, esta porção do espectro de r, também designada por **síndroma** no contexto da TCCE, pode ser usada sob certas condições para estimar o ruído já que contém uma parte da informação acerca de r. O síndroma é normalmente designado no contexto de TCCE por **assinatura** do ruído impulsivo.

O teorema da amostragem resulta de um problema de reconstrução de sinal, isto é, o problema da interpolação de um sinal $f(t)$ de banda limitada a partir das suas amostras. Pode fazer-se uma analogia entre interpolação e a correcção de apagamentos. Ou seja, quando sabemos à partida a localização dos erros, a amplitude correcta pode ser obtida através de interpolação do sinal observado nas posições correspondentes às amostras desconhecidas.

Para que seja possível a correcção de erros ou apagamentos é necessário existir redundância nos dados. Se pensarmos em termos de processamento de sinal diremos que num sinal f sobre-amostrado representado pela sua expansão

$$f(t) = \sum_{k} f(k)\phi(t-k) \tag{4.3}$$

é possível reconstruir um número finito de n amostras corrompidas a partir das correctas, resolvendo um conjunto de n equações lineares obtidas a partir da expressão (4.3),

$$f(i_j) = \sum_{k=1}^{n} f(i_k)\phi(i_j - i_k) + v_j \ ,$$

onde v_j é uma combinação linear de amostras conhecidas.

A existência de solução deste sistema deve-se à sobre-amostragem que proporciona a existência de mais amostras do que as necessárias para reconstruir o sinal e faz com que a matriz do sistema seja não-singular [Ferreira92]. No caso de não existir sobre-amostragem, o conjunto das amostras que representa o sinal é mínimo, isto é, a remoção de apenas uma torna impossível a reconstrução sem erro.

As técnicas de remoção de ruído impulsivo utilizadas em processamento de sinal podem ser interpretadas num contexto de códigos de correcção de erros sobre o corpo dos reais ou dos complexos. Foi Blahut, com os seus trabalhos [Blahut79, Blahut83, Blahut85] que estabeleceu as primeiras relações entre estas duas áreas aparentemente distintas que caminharam lado a lado ao longo de algumas décadas na segunda metade do século XX. A existência de linguagem própria em cada uma das áreas e o facto de a maior parte dos códigos de controle e correcção de erros serem implementados em **corpos finitos** (Galois Fields $GF(q)$) ao contrário das técnicas espectrais que normalmente são descritas no **corpo real ou complexo**, podem ter sido dois factores que contribuíram para esse afastamento. Vários trabalhos de outros autores se seguiram e foram desenvolvidos métodos tendo por base esta ideia [Marshall82, Marshall84, Marshall86, Kumaresan86, Marvasti99].

Vamos designar a partir daqui os códigos de correcção de erros no corpo dos reais ou dos complexos por **códigos reais**. Se estes códigos forem baseados na transformada discreta de Fourier (DFT) passam a ser designados por **códigos DFT**. No capítulo seguinte vamos relacionar e comparar estes códigos com os códigos em corpos finitos e analisar a sua estabilidade.

CAPÍTULO 5

Interpolação nos domínios do tempo e da frequência

A interpolação de sinais e imagens de banda limitada a partir de um conjunto de amostras é um tema importante da teoria de sinal. O estudo deste tipo de problemas pode ser feito no domínio do tempo como em [Marks93, Zayed93, Feichtinger94, Ferreira94C, Benedetto00], ou no domínio da frequência conforme sugerem os trabalhos [Gröchenig93A, Strohmer93, Strohmer95].

A existência de uma dualidade entre estas duas abordagens, demonstrada em [Ferreira96], pode ser utilizada para enquadrar alguns problemas num dos domínios, tempo ou frequência, de modo a que a performance para um problema específico seja óptima.

Uma das conclusões importantes acerca da existência desta dualidade relaciona-se com a possibilidade de se usarem sempre métodos de dimensão mínima para se obterem as soluções dos problemas. Ou seja, a solução sugerida em [Ferreira94B] é obtida a partir de um conjunto de equações com dimensão mínima no tempo e igual ao número de amostras desconhecidas, e as equações de Toeplitz que resolvem o sistema na frequência [Gröchenig93A, Strohmer93, Strohmer95] são também de dimensão mínima e igual ao número de elementos da DFT diferentes de zero.

A reconstrução de sinal (interpolação ou extrapolação) pode ser interpretada usando a linguagem dos códigos de controlo de erros, isto é, com códigos reais. A existência de

sobre-amostragem corresponde à introdução de redundância através de um código de controlo de erros, e o grau de redundância determina a capacidade correctora do código.

Este capítulo começa por apresentar as vantagens e desvantagens dos códigos reais. Em seguida é feita uma análise das relações e equivalências existentes entre os códigos Reed-Solomon e os códigos DFT. O mecanismo de codificação e descodificação para estes últimos é discutido e são analisados problemas de correcção de apagamentos e de erros de dimensão máxima. Ulteriormente, é proposta uma solução para o problema da reconstrução de apagamentos com códigos DFT de dimensão mínima.

5.1 Vantagens e desvantagens dos códigos reais

Uma vantagem evidente destes códigos relaciona-se com a possibilidade de serem implementados com aritmética habitual, ao contrário dos códigos em corpos finitos que normalmente necessitam de cuidados de implementação pois a sua aritmética é modular. É comum nestes últimos, o uso de processadores dedicados com o objectivo de aumentar as suas performances.

Outra das vantagens diz respeito à escolha do tamanho para o bloco de entrada e de saída do codificador (N, K). No caso dos códigos em corpos finitos, que recorde-se são normalmente implementados em GF(q) onde q é um primo (ou da forma $q=2^p$), a escolha dos parâmetros N e K é limitada e fortemente condicionada pelo valor de q. Note-se que q tem de ser tal que garanta as condições para que exista o corpo, e isto não se verifica para um valor qualquer. Nos códigos DFT esta limitação não se coloca pois a transformada de Fourier existe sempre que forem garantidas as condições para a sua existência. A existência de códigos reais de distância máxima para valores de N e K arbitrários é apresentada em [Marshall84].

Os códigos DFT são tolerantes a pequenos erros em símbolos, quando o nível de ruído é baixo [Wolf83]. Falar-se de tolerância a pequenos (ou grandes) erros não faz sentido quando temos códigos em corpos finitos. Pode no entanto associar-se a cada código uma capacidade de detectar e corrigir um número finito de erros.

Como desvantagem podemos afirmar que, contrariamente ao que acontece com os códigos em corpos finitos onde a aritmética é exacta, nos códigos reais temos sempre um erro associado aos dados devido, entre outros, aos erros de arredondamento e de representação referidos na secção 2.2. As limitações impostas à partida pela precisão

finita dos computadores ou a sensibilidade ao ruído dos algoritmos, podem provocar instabilidade numérica.

Apesar deste factor, outros como a distribuição e amplitude dos erros contribuem de uma forma mais crítica para que os algoritmos de correcção de erros ou de apagamentos sejam numericamente instáveis. Na realidade podemos verificar em [Ferreira94C, Ferreira97, Marvasti97, Rath02] que quando o padrão dos erros é do tipo rajada, as dificuldades aumentam quando se usam códigos reais para correcção de erros.

5.2 Códigos DFT e códigos Reed-Solomon

Os códigos Reed-Solomon, como já referido, são códigos por bloco que ainda hoje se utilizam em muitas aplicações. Alguns destes códigos são baseados na transformada de Fourier em $GF(q)$ e a sua capacidade de corrigir erros relaciona-se com o número de zeros consecutivos da transformada de Fourier. Na realidade, um código Reed-Solomon capaz de corrigir n erros baseia-se na existência de $2n$ zeros consecutivos da transformada de Fourier de cada palavra codificada.

É possível relacionar os códigos DFT com os códigos Reed-Solomon, admitindo que o corpo finito $GF(q)$ original é substituído pelo corpo real ou complexo e a transformada de Fourier em $GF(q)$ é substituída pela DFT. As palavras codificadas de um código DFT são limitadas em frequência no sentido de [Ferreira94C]. Considerando como na secção 4.4 o sinal observado $g=f+r$, se o sinal f for uma palavra deste código, a sua DFT $\hat{f}$ tem um número consecutivo de amostras iguais a zero, admitindo os dados originais a.

Usando uma notação algébrica podemos escrever

$$f = F^H \begin{bmatrix} a \\ 0 \end{bmatrix}. \tag{5.1}$$

onde 0 representa um vector com N-K zeros e F é a transformada de Fourier unitária definida na secção 2.1.4.

O facto de na expressão (5.1) se considerar a transformada inversa F^H e não a directa F, significa que os dados originais a estão no domínio da frequência, e a palavra codificada f está no domínio do tempo: vamos assumir este facto já que não implica perda de generalidade.

Na figura 5.1 apresenta-se um esquema da codificação de um código DFT.

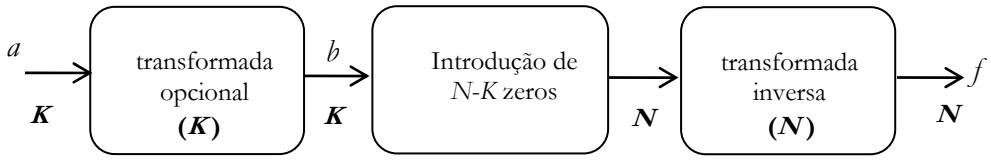

Figura 5.1: Diagrama de blocos da codificação de um código DFT

Ao sinal original *a* de tamanho *K* é aplicada uma DFT e são acrescentados zeros até se obter um vector de dimensão *N*. Do cálculo da transformada inversa de Fourier (IDFT) de tamanho *N* resulta a palavra codificada *f* cujos valores pertencem ao corpo dos complexos quando os zeros são introduzidos no fim de *a*. Como já foi referido, pode assumir-se que os dados originais estão no domínio do tempo ou da frequência. Neste sentido, a transformação associada ao primeiro bloco é claramente opcional e vamos assumir que $b=a$.

Se pretendermos que *f* seja real como é comum quando se fazem simulações, podemos acrescentar os zeros nas posições relativas às altas frequências para que exista uma simetria conjugada entre as amostras da DFT. Desta forma, a palavra *f* é equivalente a um sinal passa-baixo com *N* amostras e pode ser representado por

$$f = F^H d, \quad d = \Big[\underbrace{a_0, a_1, \ldots, a_M}_{M+1 \text{ elementos}} \overbrace{0, 0, 0, \ldots, 0}^{N-(2M+1) \text{ zeros}} \underbrace{a_M^*, a_{M-1}^*, \ldots, a_1^*}_{M \text{ elementos}}\Big]^T .$$

Nestas circunstâncias, o tamanho de *a* é sempre um número ímpar. Esta restrição não é importante e vamos assumir a partir daqui que $K=2M+1$. Quando estamos no caso complexo, *f* é constituído por $2M+1$ números complexos, e todas as palavras dum código com estas características constituem um sub-espaço de $\mathbb{C}^N$ de dimensão $2M+1$. Analogamente para o caso real, *f* pode ser representado por um vector de números reais que pertencem ao sub-espaço $\mathbb{R}^{2M+1}$ de $\mathbb{R}^N$. Normalmente designa-se este sub-espaço por **sub-espaço do código**.

Na figura 5.2 pretende-se ilustrar de uma forma vectorial o processo que muitos códigos usam para detectar e, em algumas situações, corrigir erros. As palavras do código $f_1, f_2, \ldots, f_n$ pertencem obrigatoriamente ao sub-espaço do código, isto é, estão no

mesmo plano. Quando é detectado um vector que não pertence ao plano do código, ou seja, um sinal observado *g* diferente da palavra original, existiu um erro. Um dos métodos que é usado para corrigir esse erro utiliza os conceitos de distância de códigos definidos na secção 4.2.1.1 para aproximar o sinal observado à palavra do código mais próxima de acordo com uma determinada regra.

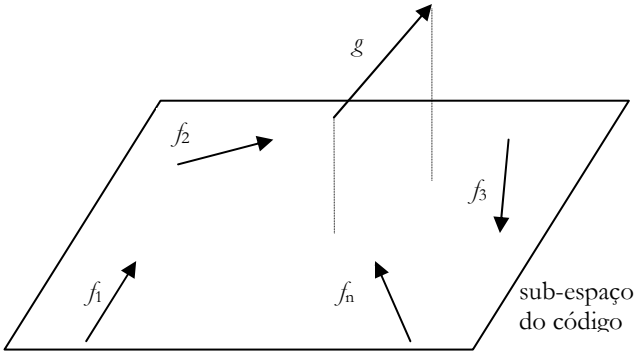

Figura 5.2: Ilustração do processo da detecção/correcção de erros. O vector *g* não pertence ao sub-espaço do código.

5.3 Códigos DFT e codificação fonte/canal em conjunto

Os sinais naturais como a voz e música ou os artificiais como os biológicos obtidos a partir de transdutores (ECG, EEC, etc.), não são em rigor limitados em frequência. No entanto, a energia da DFT destes sinais está normalmente concentrada numa determinada região do espectro, enquanto que noutra existem uma série de coeficientes de amplitude desprezável. Este facto faz com que muitos se possam aproximar com um erro arbitrariamente pequeno por um sinal limitado em frequência.

Na verdade, tomar como nulos os coeficientes de valor próximo de zero é uma técnica de codificação que surge naturalmente para este tipo de sinais quando se usa a DFT. Isto acontece quando se sabe à partida que é mantida a inteligibilidade dum sinal já que os níveis de distorção introduzidos pelo processo de codificação/descodificação são desprezáveis [Rabiner78, Kleijn95].

Os algoritmos tradicionalmente utilizados na reconstrução de sinal têm por base a limitação em frequência e a existência de sobre-amostragem que pode ser relacionada com a capacidade de correcção do algoritmo [Ferreira94C]. Como referido na secção anterior, os códigos DFT utilizam uma estratégia semelhante que, associada à codificação natural descrita no parágrafo anterior, podem constituir uma mais valia

destes códigos no que diz respeito ao tamanho do bloco. A possibilidade de codificar um sinal de *K* amostras sem aumentar o tamanho do bloco, contrasta com o processo descrito na figura 5.1 em que a criação de redundância artificial através da introdução de (*N*-K) zeros faz com que inevitavelmente o tamanho do bloco aumente.

Por outro lado, a utilização dos códigos reais nomeadamente os DFT, ajusta-se bem a esta situação já que para além do tamanho de bloco não aumentar (carga computacional menor), existe a possibilidade de ajustar a taxa de bits do código ao contrário do que acontece num código em corpos finitos onde a taxa de bits é fixa.

Assim podemos acrescentar algo à secção 5.1 em relação às vantagens dos códigos DFT. Ou seja, sendo os conceitos de conteúdo ou densidade espectral bem entendidos no corpo dos reais, a possibilidade de ajustar a taxa de transmissão de bits utilizando técnicas eficientes de codificação canal/fonte em conjunto com códigos DFT pode constituir uma mais valia quando comparada com as técnicas usuais da TCCE.

5.4 Descodificação de códigos DFT

Pode considerar-se a matriz *B* de limitação em frequência descrita na secção 2.1.2 como sendo o operador que projecta um vector de *N* amostras no sub-espaço do código

$$B = F^H \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} F \,. \tag{5.2}$$

Desta forma, cada palavra do código pode exprimir-se através da expressão $f=Bf$ como demonstra

$$f = F^H \begin{bmatrix} a \\ 0 \end{bmatrix} = F^H \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} F F^H \begin{bmatrix} a \\ 0 \end{bmatrix} = F^H \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} I \begin{bmatrix} a \\ 0 \end{bmatrix} = F^H \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ 0 \end{bmatrix} = F^H \begin{bmatrix} a \\ 0 \end{bmatrix}.$$

Normalmente em teoria espectral, designa-se por largura de banda o número de coeficientes diferentes de zero da DFT da palavra que é igual a $2M+1$. A quantidade $r=(2M+1)/N$ designa-se por taxa de sobre-amostragem. Podemos relacionar estes dois parâmetros com o operador amostragem na frequência e largura de banda normalizada definidos na secção 2.1.2. No contexto da TCCE o parâmetro de sobre-amostragem representa a taxa do código descrita na secção 4.2.1.1.

5.4.1 Correcção de apagamentos

A limitação em frequência imposta à partida para a palavra codificada *f* e o facto de se conhecerem as posições das amostras corrompidas permite concluir que existe um subconjunto de amostras de *f* que são sempre conhecidas. Se designarmos *J* como sendo um subconjunto de $\{0,1,2,\ldots,N-1\}$ não vazio constituído pelos índices dessas amostras, podemos representá-las por $\{f_i\}_{i\in J}$. Se definirmos a matriz *D* por

$$D_{ij} = \begin{cases} 1, & \text{se } i = j \text{ e } i \in J \\ 0, & \text{outros} \end{cases}, \tag{5.3}$$

podemos escrever o sinal observado *g* em função dessa matriz,

$$g = Df.$$

Corrigir então os apagamentos no conjunto complementar de *J*, consiste em obter *f* a partir de *g*. O pseudo código descrito a seguir realiza iterativamente essa tarefa.

input: *g*, sinal observado (*f* com apagamentos)
M, número de iterações
set: $f^{(0)}=g$
for *i*= 1 **to** *M* **do**
projectar $f^{(i-1)}$ no sub-espaço do código
$v=B\, f^{(i-1)}$
inserir as amostras correctas de *f* em *v*
$f^{(i)}=g + (I-D)v$
end do
output: $f^{(M)}$, aproximação de *f* após *M* iterações

Na iteração *i*, a aproximação $f^{(i-1)}$ é filtrada, ou seja, é projectada no sub-espaço do código ($v=B\, f^{(i-1)}$). Em seguida, as amostras conhecidas dadas pelo conjunto *J*, são inseridas nas posições correspondentes. Podemos reescrever este algoritmo e inserir um parâmetro de relaxação obtendo assim a expressão

$$f^{(i)} = \lambda g + (I - \lambda D)Bf^{(i-1)}. \tag{5.4}$$

Este processo iterativo é equivalente à versão discreta do algoritmo de Papoulis-Gerchberg [Papoulis75, Gerchberg74] originalmente proposta para extrapolação de sinais de banda limitada em L_2. Em [Ferreira94C] analisam-se entre

outros, limites para o erro em função do número de iterações e a taxa de convergência em função de padrões de amostras perdidas.

É possível abordar este processo iterativo de uma forma equivalente sob outras perspectivas. Em seguida apresentamos algumas com o objectivo de melhor enquadrar a utilização dos códigos DFT como mais uma área da reconstrução de sinal.

5.5 Outras interpretações do processo iterativo

O mecanismo simples de obtenção das amostras corrompidas (5.4) pode ser interpretado noutros contextos de uma forma equivalente. Os algoritmos de reconstrução propostos por [Schafer81], o método das projecções sucessivas em conjuntos convexos [Youla82] ou a reconstrução no contexto de *frames* [Pei97], contam-se entre algumas destas formas.

A utilização dos códigos DFT em correcção de erros pode também ser entendida no contexto de *frames*. O problema pode ser enunciado da seguinte forma: dada uma *frame* $X := \{B_i\}_{i\in J}$, e um conjunto de produtos internos $\langle f, B_i \rangle_{i\in J}$, reconstrua-se o símbolo original *f*. Assume-se que estando os coeficientes $\langle f, B_i \rangle_{i\in J}$ de *f* próximos dos do sinal observado *g*, pode considerar-se *f* uma boa aproximação de *g*. O algoritmo baseia-se no facto do produto interno de um vector limitado em banda *f* com uma coluna B_i de *B*, ser $\langle f, B_i \rangle = f_i$.

Mostra-se em [Ferreira99] que um conjunto de *K* colunas da matriz *B* (5.2) constituem uma *frame* para o sub-espaço dos vectores de $\mathbb{R}^N$ de banda limitada a $2M+1$ harmónicos, se $K \geq 2M+1$. Ou seja, quaisquer conjuntos de pelo menos $2M+1$ colunas $\{B_i\}_{i\in J}$ de *B*, constituem uma *frame* do sub-espaço do código (conjunto de símbolos f tais que $f=Bf$). Para estas condições, não existe nenhum $x \neq 0$ de tal forma que $x=Bx$ e $x_i=0$ para todo o $i \in J$ [Ferreira94C].

Ainda em [Ferreira99] conclui-se que a condição para constituir uma *frame* é satisfeita

$$\alpha \|f\|^2 \leq \sum_{i\in J} |\langle f, B_i \rangle|^2 \leq \beta \|f\|^2,$$

e os limites da *frame* são respectivamente o maior e menor dos valores próprios da matriz *BDB*, ou seja, $\beta=\lambda_{max}$ e $\alpha=\lambda_{min}$ (matriz *D* definida em (5.3)).

A pior situação relativamente à distribuição de *J* acontece quando ela é contígua e nesta situação α pode aproximar-se criticamente de zero. Para mais detalhes sobre a estabilidade numérica deste tipo de problemas pode consultar-se [Ferreira94D, Ferreira97, Ferreira00].

O algoritmo *frame* genérico pode definir-se por

$$f^{(n)} = f^{(n-1)} + \mu S\left(f - f^{(n-1)}\right),$$

onde *S* é o operador *frame* e $\mu = 2/(\alpha+\beta)$. Este procedimento pode ser usado para corrigir apagamentos já que *Sf* depende apenas das amostras conhecidas.

A convergência é geométrica, mas a taxa

$$\mu = \frac{\beta - \alpha}{\beta + \alpha}$$

diminui à medida que os limites da *frame* $\mu = \beta/\alpha$ aumentam. De referir que os números de condição das matrizes quando se opta pelo processo não iterativo aumenta com β/α. Ou seja, quanto maior for a razão entre o limite superior e inferior, maior é o efeito do ruído, isto é, mais problemas vão existir na recuperação das palavras corrompidas numa transmissão.

A relação entre o algoritmo de Papoulis-Gerchberg já referido, e o algoritmo das *frames* consiste na capacidade de reconstruir um sinal *f* a partir das suas projecções $\langle f, \varphi_i \rangle$ nos elementos que constituem a estrutura oblíqua.

Em [Ferreira99] é analisada em detalhe a equivalência entre todas estas abordagens e pode verificar-se como os vários factores, número de apagamentos, o padrão dos erros, a largura de banda do sinal e o valor da dimensão máxima do espaço *N*, contribuem para aproximar a palavra original *f*.

Existe algo em comum entre as várias abordagens descritas no parágrafo anterior: todos os algoritmos são de dimensão máxima, isto é, o tamanho dos vectores e das matrizes é determinado pelo número total de amostras do sinal, *N*. Quando o número de apagamentos é muito inferior a *N* esta restrição pode induzir taxas de convergência baixas, nomeadamente no caso do padrão de erros ser do tipo rajada. Para além disto, na maior parte das vezes estes algoritmos exigem mais memória, mais tempo de cálculo por iteração e em geral um número de iterações mais elevado, consequência das lentas taxas de convergência, mesmo quando se utilizam técnicas de aceleração de convergência, nomeadamente relaxação.

Apesar de existir a possibilidade de acelerar algumas tarefas destes métodos como por exemplo fazer a filtragem em hardware, o facto destes algoritmos serem de dimensão máxima pode condicionar a sua aplicação a problemas específicos com imposições de tempo real curtos, como é o caso de voz sobre IP ou *streaming* de vídeo. Na secção seguinte propõem-se duas estratégias que conduzem a algoritmos que vamos passar a designar de **dimensão mínima no tempo** ou **dimensão mínima na frequência**.

5.6 Algoritmos de dimensão mínima

Quando se utilizam técnicas iterativas apropriadas, as taxas de convergência dos métodos que têm ordem inferior à máxima, são regra geral, muito superiores às atingidas pelos métodos de dimensão máxima.

Ambas as formulações como já foi referido, conduzem a um conjunto de equações lineares de dimensão inferior à máxima do espaço que é *N*. No domínio do tempo o conjunto de equações é igual ao número de amostras desconhecidas $\{f_i\}_{i\in\bar{J}}$ considerando que $\bar{J}$ representa o conjunto complementar do conjunto dos índices das amostras conhecidas J em $\{0,1,2,\ldots,N-1\}$. O conjunto de equações na frequência é também de dimensão mínima e de número igual aos coeficientes diferentes de zero da DFT de f (a dimensão do sub-espaço do código).

O tamanho e a estrutura das equações obtidas por estas duas formulações são naturalmente diferentes. No entanto, como de ambas resulta um sistema de equações lineares, a solução pode obter-se nos domínios do tempo ou da frequência utilizando as técnicas iterativas, directas ou semi-iterativas referentes aos métodos descritos no capítulo três. No capítulo seguinte, são comparados os desempenhos de alguns destes algoritmos para um conjunto de problemas, no sentido de identificar qual dos métodos melhor se ajusta a um problema específico, já que concluímos a não existência de um método ideal que resolve todos os problemas de uma forma óptima. Para já vamos apresentar as formulações associadas a cada um dos domínios.

5.6.1 Formulação das equações de dimensão mínima no tempo

Vamos descrever o procedimento que leva à formulação do conjunto de *n* equações para as amostras desconhecidas $\{f_i\}_{i\in\bar{J}}$, considerando que o seu cardinal é *n*. Se

associarmos os factos de B ser uma matriz de limitação em frequência e da palavra codificada satisfazer a $f=Bf$, podemos concluir que existe um conjunto de amostras da DFT de f que são conhecidas e iguais a zero. Vamos designar por J' o conjunto constituído pelos índices das amostras conhecidas na frequência e por $\overline{J}'$ o seu complemento em $\{0,1,2,\ldots,N-1\}$.

Partindo da equação $f=Bf$ podemos obter um conjunto de n equações igual ao número de amostras desconhecidas $\{f_i\}_{i\in\bar{J}}$ decompondo a equação numa adição

$$f_i = \sum_{j\in J} B_{ij} f_j + \sum_{j\in J} B_{ij} f_j \quad , i \in \bar{J} .$$

Na forma matricial pode escrever-se

$$y = P y + c ,$$

onde a solução desta equação é o vector dos apagamentos y. O vector c,

$$c_i = \sum_{j\in J} B_{ij} f_j \quad , i \in \bar{J} ,$$

tem dimensão n e é conhecido à partida, pois depende apenas de quantidades conhecidas.

A matriz P de dimensão $n \times n$ é uma sub-matriz principal de B, que se obtém eliminando as linhas e colunas dadas pelos índices de J. Vamos usar a notação

$$P = B(\bar{J}, \bar{J})$$

já que podemos construir a matriz P a partir de B seleccionando as linhas e colunas dadas precisamente pelos elementos do conjunto $\bar{J}$. Cada elemento da matriz B de dimensão $N\times N$ definida em (5.2) pode ser obtido pela expressão

$$B_{jk} = \sum_{l\in\overline{J}'} F^*_{lj} F_{lk} = \tfrac{1}{N} \sum_{l\in\overline{J}'} e^{\mathrm{j}\frac{2\pi}{N}(j-k)l} .$$

Ao contrário da dimensão máxima referida para os métodos na secção anterior, a dimensão da matriz P, e por inerência a do problema, é mínima e igual ao número de amostras desconhecidas n.

Em geral, a matriz P é Hermítica e não-negativa definida. Prova-se em [Ferreira94B, Ferreira96] que quando o número de apagamentos n é pequeno, mais precisamente quando o número de amostras conhecidas $\{f_i\}_{i\in J} \geq 2M+1$, a matriz $(I - P)$ é positiva definida e $\rho(P)<1$. Apesar desta ser uma condição necessária para se obter uma solução

para o sistema $(I–P)y=c$, de n equações, pode não ser suficiente na prática pois podem existir outros problemas associados, como já foi referido na secção 2.2.

No capítulo seguinte é feita uma análise em detalhe de diferentes métodos para a resolução destes sistemas no domínio do tempo onde são eleitos os algoritmos que melhor se adaptam a problemas específicos em função dos recursos de hardware disponíveis.

5.6.2 Formulação das equações de dimensão mínima na frequência

Vamos descrever agora o procedimento que conduz ao conjunto de equações de dimensão mínima no domínio da frequência. Como já foi referido, sendo f uma palavra codificada de banda limitada faz com que a sua DFT $\hat{f} = Ff$ tenha um conjunto de amostras iguais a zero cujos índices são dados pelos elementos do conjunto J'. O conjunto de amostras da DFT conhecidas podem representar-se por $\{\hat{f}_i\}_{i\in J'}$. De notar que ao contrário do que acontece no caso da formulação no tempo, neste caso as amostras conhecidas são todas iguais a zero.

O complementar do conjunto J' tem dimensão $2M+1$ e designa-se por $\overline{J}'$. Ou seja, a palavra codificada é completamente descrita pelas $2M+1$ amostras $\{\hat{f}_i\}_{i\in \overline{J}'}$.

Para obter o conjunto de $2M+1$ equações para estas quantidades desconhecidas necessitamos duma equação no domínio da frequência análoga à $f=Bf$ usada na secção anterior. Para isso vamos considerar a seguinte equivalência

$$\hat{f} = Ff = F(I-D)f + FDf = F(I-D)F^H \hat{f} + \hat{v}\,.$$

O vector $\hat{v}$ depende apenas de quantidades conhecidas e pode ser obtido calculando a DFT do vector obtido após a substituição por zero das amostras de f desconhecidas. Tanto esta formulação como a da secção anterior são analisadas em detalhe em [Ferreira96].

Vamos definir a matriz $C = F(I-D)F^H$. Sendo assim, temos $\hat{f} = C\hat{f} + \hat{v}$, e podemos decompor a equação da seguinte forma

$$\hat{f}_i = \sum_{j\in\overline{J}'} C_{ij}\hat{f}_j + \sum_{j\in J'} C_{ij}\hat{f}_j + \hat{v}_i\,.$$

A parcela do meio desta adição é igual a zero, já que $\hat{f}_j = 0, \quad j \in J'$. Se limitarmos o índice *i* ao conjunto das amostras da DFT desconhecidas $\overline{J}'$, obtemos um conjunto de 2*M*+1 equações

$$\hat{f}_i = \sum_{j \in \overline{J}'} C_{ij} \hat{f}_j + \hat{v}_i, \quad i \in \overline{J}'.$$

Temos então na forma matricial

$$\underline{z} = \underline{\underline{Q}}\, \underline{z} + \underline{d},$$

onde $\underline{z}$ representa o vector das amostras da DFT desconhecidas e tem dimensão 2*M*+1. Os 2*M*+1 elementos do vector *d* são conhecidos pois constituem um subconjunto dos elementos de $\hat{v}$ igual a

$$d_i = \sum_{j \in J} F_{ij} f_j, \quad i \in \overline{J}'.$$

A matriz $\underline{\underline{Q}}$ de dimensão (2*M*+1) × (2*M*+1) é uma sub-matriz principal de *C*, que se obtém eliminando as linhas e colunas dadas pelos índices de J'. Vamos usar novamente a notação

$$\underline{\underline{Q}} = C\left(\overline{J}', \overline{J}'\right),$$

já que podemos construir a matriz $\underline{\underline{Q}}$ a partir de *C* seleccionando as linhas e colunas dadas precisamente pelos elementos do conjunto $\overline{J}'$. Cada elemento da matriz *C* é dado por

$$C_{jk} = \sum_{l \in \overline{J}} F_{jl} F_{kl}^{*} = \frac{1}{N} \sum_{l \in \overline{J}} e^{-\mathrm{j} \frac{2\pi}{N}(j-k)l}.$$

Em geral, $\underline{\underline{Q}}$ é uma matriz complexa, não-negativa definida e Hermítica. O seu tamanho é independente do número de apagamentos e depende apenas da largura de banda da palavra codificada. Para além disto, quando o número de amostras conhecidas $\{f_i\}_{i \in J} \geq 2M+1$, a matriz $I-\underline{\underline{Q}}$ é positiva definida e $\rho(\underline{\underline{Q}})<1$ [Ferreira94B, Ferreira96].

O facto das amostras conhecidas na frequência serem todas nulas contrariamente ao que acontece às que se conhecem no tempo, faz com que existam assimetrias entre as duas formulações agora apresentadas. Para além das diferenças óbvias, a aritmética complexa dos métodos do domínio da frequência com os custos computacionais associados e referidos na secção 2.2, surge em consequência dessa assimetria.

No entanto, como ambas as formulações conduzem a sistemas de equações lineares, a opção entre sistemas de dimensão mínima no tempo ou na frequência pode ser feita em

função do desempenho de cada método. No capítulo seguinte, vamos ainda mais longe ao estabelecer os pontos críticos que naturalmente seleccionam os métodos mais adequados a cada problema.

Na tabela 5.1 apresenta-se em resumo, a notação envolvida nas duas formulações acima descritas.

ALGORITMOS DE DIMENSÃO MÍNIMA

TEMPO		FREQUÊNCIA
$f_i,\ i \in J$	Amostras conhecidas	$\hat{f}_i,\ i \in J'$
$f_i,\ i \in \bar{J}$	Amostras desconhecidas	$\hat{f}_i,\ i \in \overline{J}'$
$n = \text{card}(\bar{J})$	Número amostras desconhecidas	$\text{card}(\overline{J}') = 2M+1$
$y = Py + c$	Equação principal	$\underset{\sim}{z} = \underset{\sim}{Q}\underset{\sim}{z} + d$
$c_i = \sum_{j \in J} B_{ij} f_j$	Vector	$d_i = \sum_{j \in J'} C_{ij} \hat{f}_j$
$P = B(\bar{J}, \bar{J})$	Matriz	$\underset{\sim}{Q} = C(\overline{J}', \overline{J}')$
$B = F^H(I - D')F$	Matriz	$C = F(I - D)F^H$
$D'_{ii} = \begin{cases} 1, & i \in J' \\ 0, & \text{----} \end{cases}$	Matriz	$D_{ii} = \begin{cases} 1, & i \in J \\ 0, & \text{----} \end{cases}$

Tabela 5.1: Sumário das formulações no tempo e na frequência. O vector y, solução do sistema no tempo, determina os n apagamentos e por consequência a palavra codificada. A solução $\underset{\sim}{z}$ para as equações na frequência é constituída pelas amostras da DFT da palavra do código.

5.7 Correcção de erros

Quando se desconhece por completo a posição das amostras erradas (que define o chamado "padrão de erros"), é impossível utilizar a iteração (5.4) para recuperar as amostras perdidas, já que o passo referente à actualização da informação que se conhece requer o conhecimento acerca das posições onde se perderam as amostras. Como é referido na secção 4.4, tipicamente, a nova estratégia consiste em dividir o problema em duas tarefas ordenadas: primeiro detectam-se os erros fazendo uso de técnicas não lineares e em seguida corrigem-se nessas posições as amplitudes erradas.

O sinal observado g é igual à soma do símbolo f e o ruído r, logo a sua DFT satisfaz a

$$\hat{g} = \hat{f} + \hat{r}\,.$$

No entanto, sendo *f* um sinal limitado em banda, para o conjunto de índices J', temos $\hat{f}_i = 0,\ i \in J'$. Logo, podemos concluir que a DFT do sinal de erro ou síndroma (secção 4.5) é parcialmente conhecida, nomeadamente para os elementos de J', ou seja,

$$\hat{r}_i = \hat{g}_i,\ i \in J'.$$

Podemos então admitir que o problema da correcção de erros é equivalente ao problema de extrapolar *r*. A seguir apresenta-se um processo iterativo que encontra dentro de certos limites a posição dos erros fazendo uso da amplitude das amostras erradas [Vieira97].

input: *g*, sinal observado (*f* com erros, *g*=*f*+*r*)
M, número de iterações

set: $\hat{b}^{(0)} = 0$ (por exemplo)

for *i*= 1 **to** *M* **do**

inserir as amostras correctas de $\hat{r}$ em $\hat{b}^{(i-1)}$

IFFT : $b^{(i)} = F^{-1}\hat{b}^{(i-1)}$

escolha de um valor *L* para o *threshold*

for *j*= 0 **to** *N*-1 **do**

if $\left|b_j^{(i)}\right| < L$ **then**

$\left|b_j^{(i)}\right| = 0$

end if

end do

FFT: $\hat{b}^{(i)} = Fb^{(i)}$

end do

output: $b^{(M)}$, aproximação de *r* após *M* iterações

Neste método assume-se que *r* é esparso, ou seja, contém muitos zeros. Assim na iteração *M*, os índices associados aos valores não nulos de *b* correspondem à localização dos erros.

O número máximo de erros que podem ser corrigidos é igual a metade do número de zeros da DFT de *f*. No caso de apagamentos e para os métodos referidos anteriormente

equivalentes à iteração (5.4), esse valor aumenta para o máximo, ou seja, é igual ao número total de zeros da DFT.

Note-se a diferença por um factor de dois entre a correcção e a detecção de erros. A diferença entre os valores limites teóricos para os dois problemas deve-se também ao facto de no caso da correcção de erros se desconhecerem a posição e as amplitudes das amostras desconhecidas.

O desempenho desta iteração é satisfatório se o número de erros for bem inferior ao limite teórico ou se o padrão de erros não for contínuo. Neste caso, conseguem-se boas aproximações para um número de iterações entre 10 e 20 [Vieira97].

Situações mais complicadas normalmente resolvem-se utilizando técnicas não lineares não iterativas. Em [Vieira00] faz-se uma descrição detalhada de alguns destes algoritmos entre os quais o método de Prony na sua versão original [Prony85].

CAPÍTULO 6

Descodificação de códigos reais com métodos de dimensão mínima

Vimos no capítulo cinco que a descodificação de códigos reais pode ser obtida resolvendo um certo conjunto de equações lineares como um dos passos. Estas equações podem obter-se utilizando duas classes de métodos, que designámos de dimensão mínima no tempo e dimensão mínima na frequência. Nos primeiros, ao encontrarmos a solução, obtemos uma aproximação das amostras desconhecidas do sinal (n), enquanto que nos segundos são os coeficientes da DFT do sinal que são estimados ($2M+1$). Apesar de existirem variantes multi-canal com vantagens interessantes [Ferreira03] vamos, no âmbito deste trabalho, analisar apenas problemas de descodificação de códigos DFT de um canal com correcção de apagamentos.

No projecto dum algoritmo a estrutura da matriz é um factor de extrema importância. No nosso caso, no domínio do tempo a matriz é Toeplitz quando as amostras perdidas são contíguas, mas para padrões de erros irregulares essa estrutura não existe. As equações na frequência são Toeplitz, pois as amostras conhecidas na frequência são sempre contíguas (sinais passa-baixo). A aceleração de Toeplitz (multiplicações vectoriais e matriciais usando FFT) pode ser realizada quando se calcula a solução na frequência. De notar que a convergência é condicionada não só pela estrutura da matriz, mas também pelo seu número

de condição. Considere-se uma matriz A diagonal mas com um elemento muito pequeno: $a_{ii}=\varepsilon$, $a_{jj}=1_{j\neq i}$. A estrutura diagonal da matriz torna a inversão trivial, do ponto de vista teórico. Na prática, se ε for suficientemente pequeno a matriz não terá inversa.

De notar que as equações do domínio da frequência usam aritmética complexa, o que aumenta a complexidade nos cálculos. Apesar disto, regra geral se $2M+1 << n$ deve optar-se por métodos na frequência em detrimento dos no tempo. Se $2M+1 >> n$ os métodos no tempo devem ser os eleitos.

Cedo se tornou claro que não existia o método ideal que resolvesse todos os problemas. No entanto, os testes a seguir apresentados provam que seguindo algumas regras podemos encontrar uma classe de métodos óptimos (estabilidade/eficiência) para uma dada aplicação.

6.1 Conjunto completo de métodos

Classificámos no capítulo três os métodos para a resolução de sistemas lineares em directos, semi-iterativos e iterativos. É possível agora estabelecer uma nova ordem aos diversos métodos apresentados tendo em conta a dimensão dos espaços associados e o facto de terem como domínio o tempo ou a frequência. (tabela 6.1).

	TEMPO	**FREQUÊNCIA**
DIRECTOS	CHT	CHF
	LEVT (*)	LEVF
SEMI-ITERATIVOS	CGT	CGF
	CGTT (*)	CGFT
ITERATIVOS	SIT	SIF
	SITT (*)	SIFT
	SORT	SORF
	PG	

Tabela 6.1: Conjunto completo de métodos para a resolução de sistemas de equações lineares. CH_- *Cholesky*; LEV_- *Levinson*; CG_- Gradientes Conjugados; CGT_- Conjugados Gradientes *Toeplitz*; SI_-Iteração simples; SIT_-Iteração simples *Toeplitz*; SOR_- *Sucessive Over-Relaxation*; PG- *Papoulis Gerchberg* (*frame method*); (*) Nem sempre aplicável.

Esta perspectiva pode clarificar algumas ligações entre os diferentes algoritmos e permite completar o conjunto de métodos com todas as formulações consideradas no tempo e na frequência. Refira-se que o termo "**completar**" deve entender-se no sentido seguinte: para um dado conjunto de métodos iterativos, directos e semi-iterativos e para as formulações do problema da interpolação no tempo e frequência apresentadas no capítulo cinco, todas as possibilidades são esgotadas.

As matrizes associadas ao sistema de equações têm, no caso do domínio do tempo dimensão n, onde n representa o número de amostras perdidas, e no domínio da frequência $2M+1$, onde M diz respeito aos harmónicos desconhecidos do sinal. O algoritmo de Papoulis-Gerchberg (*frame method*) serve de comparação com as duas classes de métodos apresentados. Neste método, em cada iteração existem duas projecções alternadas, uma no domínio da frequência e outra no domínio do tempo. É de dimensão máxima, já que manipula vectores da mesma dimensão do espaço onde está definido o problema, N. Em todos os outros casos, a dimensão é mínima já que coincide com a dimensão do sub-espaço das amostras desconhecidas.

6.2 Caracterização das experiências

O conhecimento acerca da estabilidade e eficiência dos diferentes métodos numa determinada máquina pode revelar-se importante se decidirmos escolher o melhor método a aplicar num determinado problema de correcção de erros utilizando códigos DFT. A comparação destas características entre sistemas semelhantes onde o que varia são recursos de hardware (tabela 6.2), pode permitir também que em certas situações as limitações impostas pela máquina não sejam de todo impeditivas para a obtenção dos melhores resultados.

	Modelo	CPU (MHz)	Cache (KiB)	RAM (MiB)
Máquina_1	*Pentium* II (*Deuchutes*)	392.021	512	320
Máquina_2	AMD-K6(tm) 3D *processor*	350.798	64	256
Máquina_3	*Celeron* (*Mendocino*)	525.017	128	64
Máquina_4	HP XE_2 *Pentium* III	450	256	128
Máquina_5	AMD *Duron* (tm) *processor*	799.623	64	128
Máquina_6	*Pentium* IV	2533.346	512	512/1024

Tabela 6.2: Características das várias arquitecturas.

Criou-se um conjunto de experiências que se executaram em todas as máquinas com o objectivo de extrair conclusões que permitam avaliar os vários algoritmos em diferentes situações. Reunida essa informação devemos ser capazes de responder quantitativamente e qualitativamente a questões, como por exemplo:

- Para uma determinada taxa do código qual o valor do número de amostras desconhecidas n que fará optar por métodos na frequência em detrimento dos do domínio do tempo, para um determinado padrão de erros? Quais os pontos críticos?
- Até que ponto a natureza do padrão de erros (contígua, aleatória, intervalos regulares) afecta os resultados?
- Qual a influência da taxa do código, $r=(2M+1)/N$, na localização dos pontos críticos?
- Qual o impacto do padrão de erros na estrutura das matrizes e respectiva repercussão na eficácia dos algoritmos?
- Qual a relação entre o aumento da capacidade computacional e o tempo necessário para encontrar a solução dos vários métodos?
- Em que medida o número de processadores numa máquina altera o desempenho global?
- Qual a influência do tipo de memórias (*cache* ou RAM) nos resultados?

6.2.1 Escolha dos parâmetros

Cada experiência é definida pelos seguintes parâmetros:

- tamanho máximo do bloco, N;
- harmónicos desconhecidos, $2M+1$;
- amostras desconhecidas no tempo, n;
- padrão de erros.

Como se pretende avaliar de que forma um determinado sistema é sensível à natureza de um problema, pareceu-nos razoável ao longo de todas as experiências manter alguns parâmetros constantes e estabelecer limites para outros. Assim para uma comparação mais consistente admite-se como fixos o valor do erro final pretendido igual a 10^{-8}, e o limite de

tempo máximo para a execução de cada método (iterativo). O erro considerado é o dado pela expressão

$$TSE = \sum_{i=1}^{N} |e_i|^2 \ .$$

Queremos obter conclusões que permitam, por um lado, avaliar os métodos em cada máquina, e por outro, comparar os resultados com os obtidos em outras máquinas semelhantes. Neste sentido, e em relação às unidades de tempo, vamos relacioná-las com o tempo de uma iteração do algoritmo de Papoulis-Gerchberg. Ou seja, desprezando o tempo de repor as amostras que se conhecem, vamos considerar que 1 unidade de máquina (**1u**) é igual ao tempo de filtrar um sinal com N amostras usando a FFT nessa máquina (tabela 6.3).

	Modelo	CPU (MHz)	**1u** (ms)
Máquina_1	*Pentium* II (*Deuchutes*)	392.021	≈ 100
Máquina_2	AMD-K6(tm) 3D *processor*	350.798	≈ 5.4
Máquina_3	*Celeron* (*Mendocino*)	525.017	≈ 1.8
Máquina_4	HP XE_2 *Pentium* III	450	≈ 1.3
Máquina_5	AMD *Duron* (tm) *processor*	799.623	≈ 0.6
Máquina_6	*Pentium* IV	2533.346	≈ 0.33

Tabela 6.3: Valor de **1u** para cada máquina.

Para além de ser mais independente do hardware, esta opção permite ter uma ideia imediata do custo computacional de cada algoritmo face ao de uma filtragem com FFT. Em todas as rotinas e para todas as máquinas o tempo limite de execução é 8000 unidades de tempo (**8000u**).

O tamanho máximo do bloco é sempre igual a N=4096, pois como o tempo por iteração varia de forma não linear com N, $\vartheta(N\log N)$, comparações com diferentes tamanhos de bloco devem ser feitas com cuidado.

Os valores limites para M e n são respectivamente [10, 500] e [8, 400]. A escolha destes limites pretende, por um lado, fazer com que quando M=10 os valores do erro final e tempo obtidos para os métodos na frequência sejam comparáveis em relação aos obtidos para a classe no tempo quando n=8, e por outro, que seja possível encontrar na área de

interesse, a sombreado da figura 6.1, os pontos críticos a partir dos quais se seleccionam naturalmente os métodos.

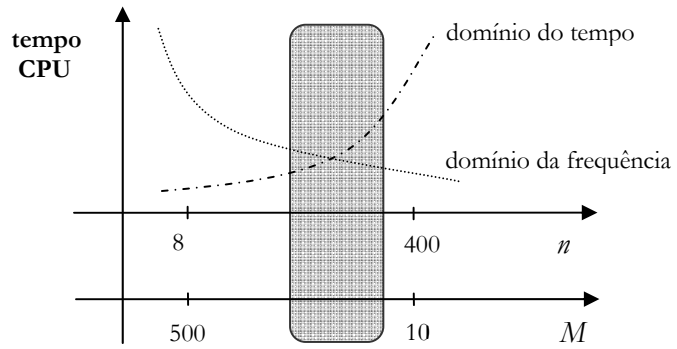

Figura 6.1: Pesquisa dos pontos críticos.

Como foi já referido, neste trabalho não se discutem as questões relativas aos blocos de modulação e desmodulação que se encontram dentro do tracejado na figura 6.2. Esta figura descreve em traços gerais o modelo usado para a obtenção dos resultados.

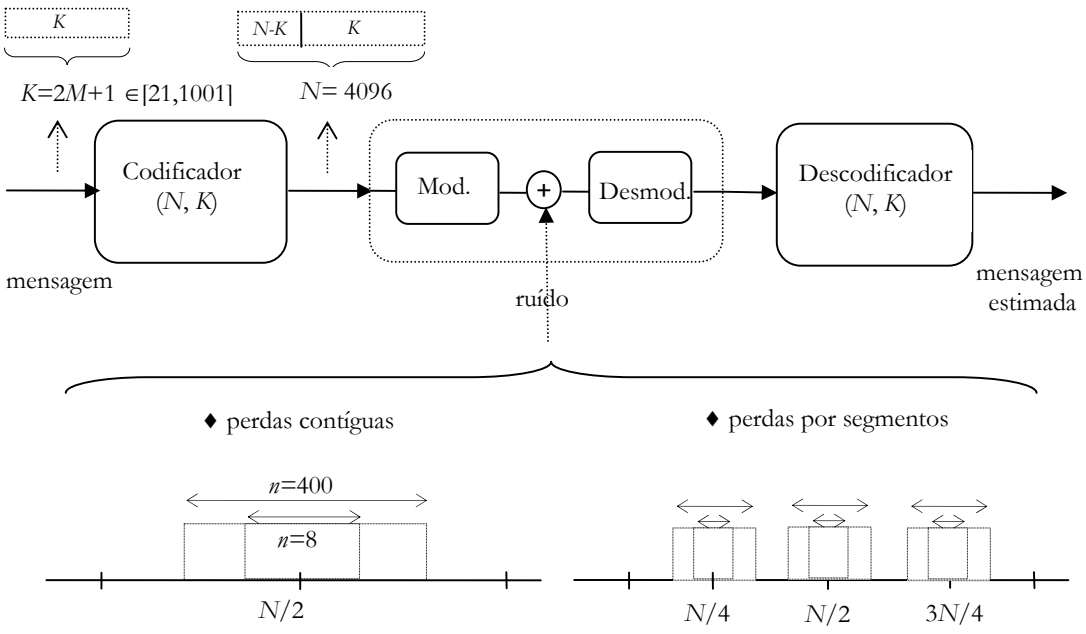

Figura 6.2: Esquema de codificação/descodificação com códigos DFT utilizado para as simulações.

Ao fazermos variar o número de harmónicos desconhecidos, estamos a considerar que a taxa do código, $r=(2M+1)/N$, toma valores dentro do intervalo [0.005, 0.25], mantendo a simetria e carácter real dos dados.

O padrão de erros, traduzido pelo ruído introduzido pelo canal nos dados pode ser de três tipos: em amostras contíguas, em segmentos, ou de carácter aleatório. A distribuição das amostras perdidas quando as perdas são contíguas acontece no meio do bloco de dados, e quando são por segmentos ocorre em zonas correspondentes a metade, um quarto e três quartos do tamanho máximo dos dados como podemos observar na figura 6.2. No caso de perdas aleatórias, força-se a distribuição a não cair num dos casos anteriores através da geração de posições baseadas no conceito de distância circular mínima e máxima entre amostras perdidas.

No sentido de manter a análise consistente, em todas as máquinas utilizou-se o compilador para Linux gcc versão 2.96. No cálculo das transformadas directa e inversa de Fourier são usadas as rotinas da FFTW [Frigo99], previamente compiladas em cada máquina quer em precisão simples quer em dupla no sentido de serem obtidas as melhores performances em relação ao número de MFLOPS (apêndice B).

Assim em resumo, definiram-se problemas tipo de teste que se executaram em todas as máquinas da tabela 6.2: Problema_1, Problema_2, Problema_3 e Problema_4. Para cada um dos problemas alguns parâmetros são mantidos constantes enquanto que outros variam. Para além disto, consideram-se três tipos de padrão de erros em cada problema, contíguo **(c)**, segmentos **(s)** e aleatório **(a)**, o que totaliza doze casos, figura 6.3.

♦ **Problema_1**	♦ **Problema_2**	♦ **Problema_3**	♦ **Problema_4**
N=4096 M=500 $\boldsymbol{n \in [8, 400]}$ **(c); (s); (a)**	N=4096 M=10 $\boldsymbol{n \in [8, 400]}$ **(c); (s); (a)**	N=4096 n=8 $\boldsymbol{M \in [500, 10]}$ **(c); (s); (a)**	N=4096 n=400 $\boldsymbol{M \in [500, 10]}$ **(c); (s); (a)**

Figura 6.3: Especificação dos problemas de teste.

Como discutiremos mais à frente, os resultados obtidos pelos Problema_1 e Problema_3, mostram claramente que os métodos do tempo têm regra geral um comportamento mais eficiente do que os na frequência. Já no que diz respeito ao Problema_4 as conclusões são opostas, agora com os métodos na frequência a evidenciar um desempenho melhor do que os no tempo, nomeadamente quando o código é mais redundante (M=10). Por outro lado, a localização dos pontos críticos em cada máquina e a comparação com máquinas distintas (tabela 6.2) pode ser esclarecedora quando se analisam os resultados obtidos no Problema_2.

6.3 Comentários sobre os problemas

6.3.1 Melhores desempenhos no domínio do tempo (Problema_1)

Considerando o tamanho máximo do bloco já referido N=4096, vamos admitir um código com taxa de $r \approx 0.25$ (M=500). Mantendo estes valores e fazendo variar a quantidade de amostras perdidas contíguas no tempo, verifica-se que esse valor não pode ser elevado, pois a partir de aproximadamente 0.5% do número total de amostras (≈ 20), praticamente todos os métodos iterativos quer do tempo quer da frequência atingem rapidamente o limite máximo de 8000u para o tempo de execução, com um consequente impacto negativo na aproximação desejada (figuras 6.4, 6.5).

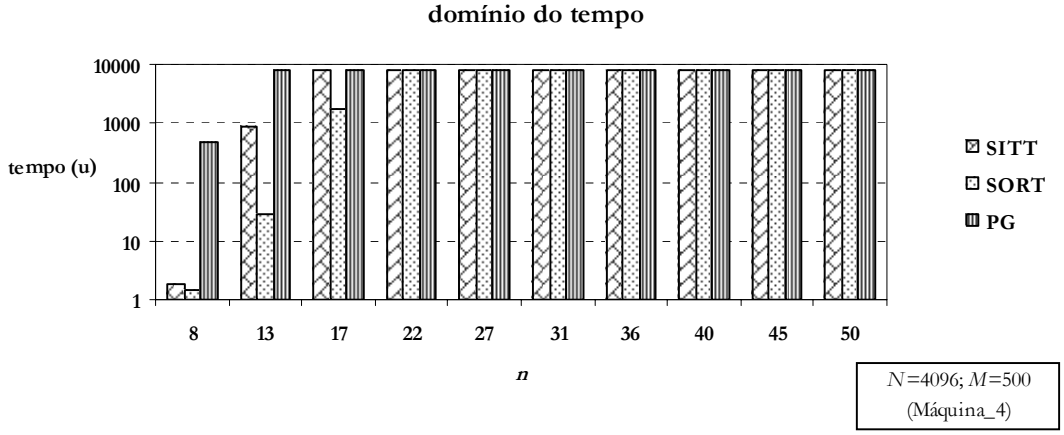

Figura 6.4: Métodos iterativos no tempo para perdas contíguas no Problema_1.

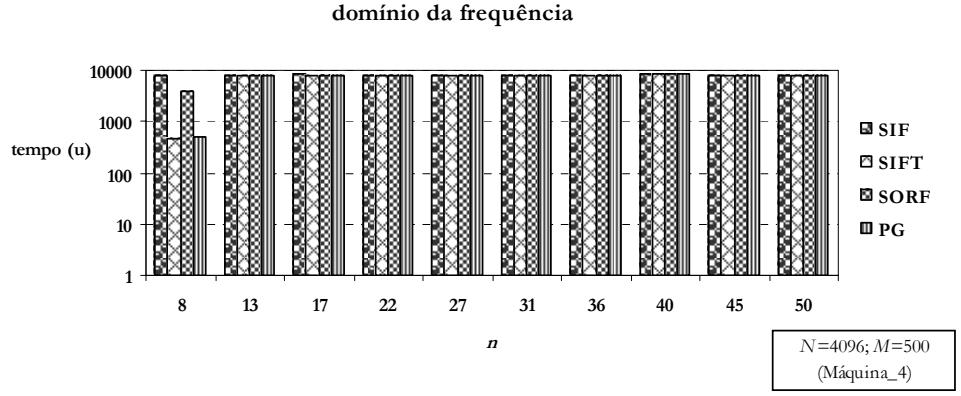

Figura 6.5: Métodos iterativos na frequência para perdas contíguas no Problema_1.

Neste caso o problema é mal condicionado o que implica números de condição elevados associados às matrizes dos métodos. No caso dos métodos iterativos, isto explica a lenta convergência.

Como M=500, o número de harmónicos desconhecidos e logo o número de incógnitas que os métodos do domínio da frequência encontram é muito grande ($2M$+1=1001), o que faz com que a reconstrução da palavra do código se torne mais difícil nesse domínio. Se os compararmos com os métodos directos do tempo, podemos observar nas figuras 6.6 e 6.7 que os do domínio do tempo são aproximadamente dez vezes mais rápidos: ≈ 1**u** para FCHT e LEVT, e ≈ 100**u** para FCHF e LEVF.

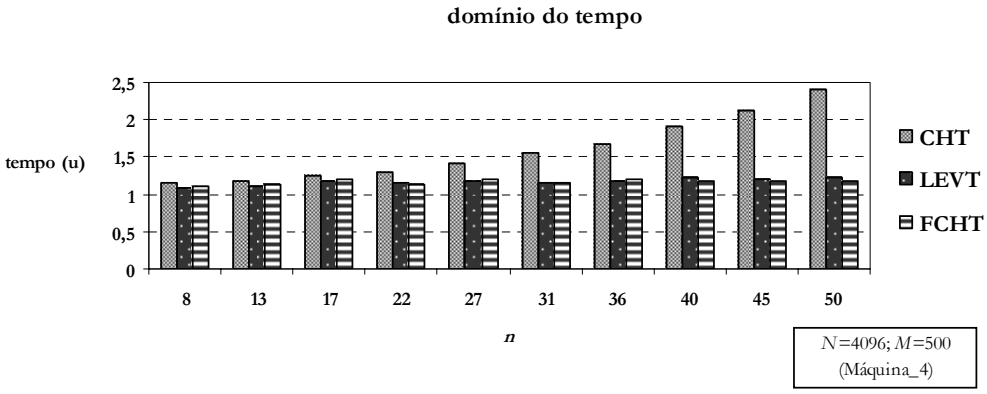

Figura 6.6: Métodos directos no tempo para perdas contíguas no Problema_1.

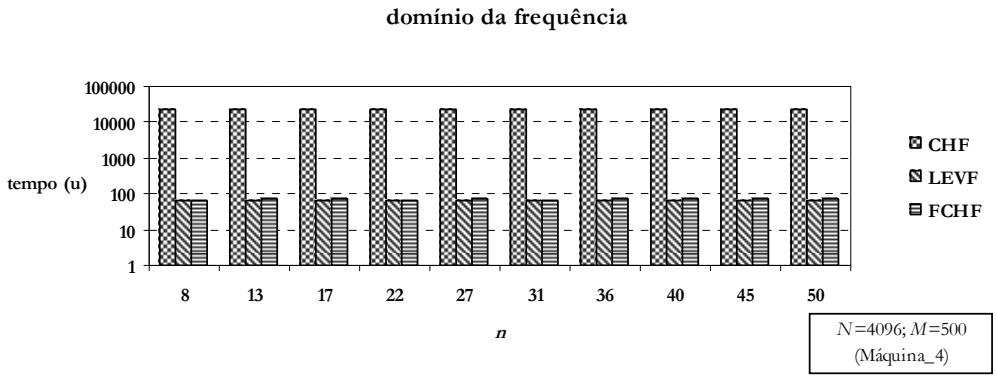

Figura 6.7: Métodos directos na frequência para perdas contíguas no Problema_1.

Como seria de esperar, sendo $n<<2M+1$, o método óptimo será eleito a partir da classe dos métodos no tempo. Se admitirmos que a decomposição de Cholesky pode ser pré-calculada, temos o FCHT e LEVT como os que apresentam melhores performances. Um método que tem um comportamento análogo apesar de ter como desvantagem começar a tornar-se instável um pouco mais cedo, figura 6.8, é o baseado em gradientes conjugados com possibilidade de ser acelerado devido à matriz ser Toeplitz, CGTT.

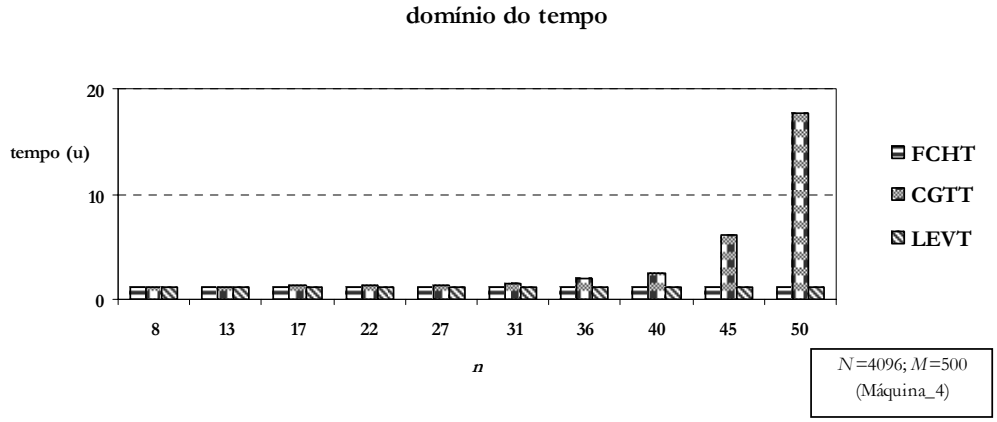

Figura 6.8: Métodos óptimos no tempo para Problema_1 quando as perdas são contíguas.

Como já foi mencionado, as matrizes dos métodos directos na frequência são sempre Toeplitz, uma vez que consideramos sempre sinais do tipo passa-baixo. Por isso podemos usar sempre o método de Levinson (LEVF). No caso dos métodos do domínio do tempo, as matrizes só são Toeplitz quando as amostras desconhecidas são equidistantes, e só nessas circunstâncias é que se pode aplicar o método LEVT.

Se agora o padrão de erros for por segmentos, as matrizes no domínio do tempo deixam de ter uma estrutura Toeplitz, logo os métodos que utilizam esta aceleração deixam de ser aplicáveis. Na frequência a estrutura mantém-se, mas como a quantidade de harmónicos é elevada os ganhos não são significativos. Apesar disto, o número de amostras perdidas pode como seria de esperar, ser um pouco mais elevado em comparação com perdas contíguas, $\approx 5\% N$ ($\approx$ 200), figura 6.9. Para este caso, um método com bom desempenho seria o FCHT.

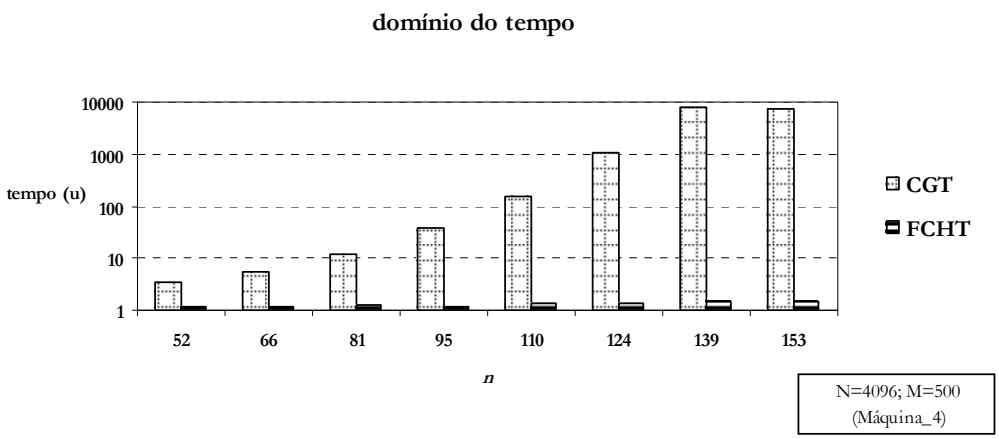

Figura 6.9: Métodos óptimos no tempo para Problema_1 quando as perdas são por segmentos.

Podemos demonstrar com este problema que a afirmação da não existência dum método ideal que resolva todas os problemas é um facto. Senão vejamos: contrariamente ao que acontece para outras distribuições de erros, agora quando o padrão de erros é aleatório podemos ver nas figuras 6.10 e 6.11 o comportamento do método PG que se revela um dos melhores à medida que o número de amostras perdidas aumenta.

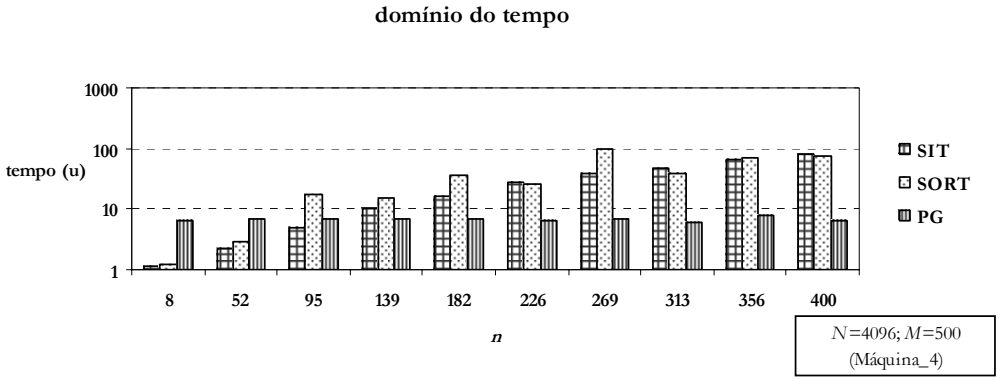

Figura 6.10: Métodos iterativos no tempo e PG para Problema_1 quando o padrão de erros é aleatório.

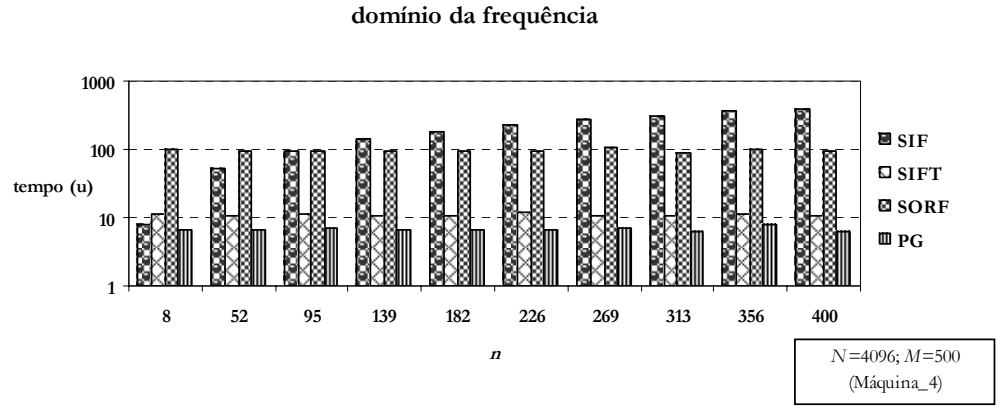

Figura 6.11: Métodos iterativos na frequência e PG para Problema_1 quando o padrão de erros é aleatório.

Nas figuras 6.12 e 6.13 podemos confirmar as performances do PG agora quando comparado com outros métodos e concluir que apesar deste problema ser mal condicionado como já referido dificultando a tarefa de qualquer método, os do domínio do tempo têm regra geral um melhor comportamento.

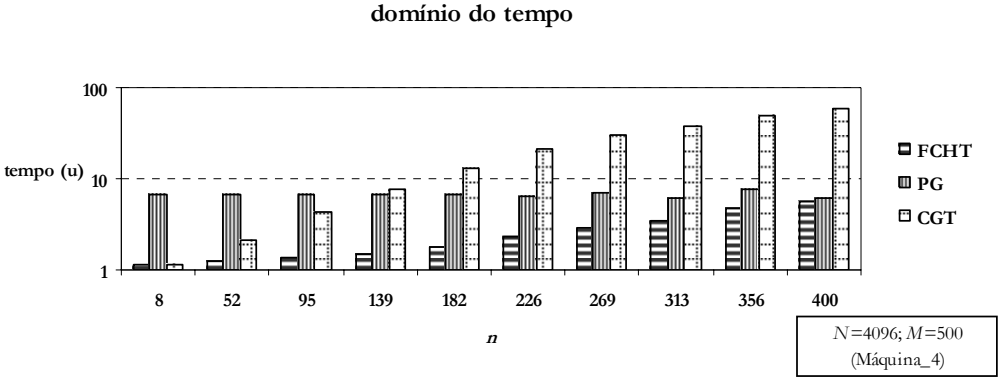

Figura 6.12: Métodos no tempo e PG para Problema_1 quando o padrão de erros é aleatório.

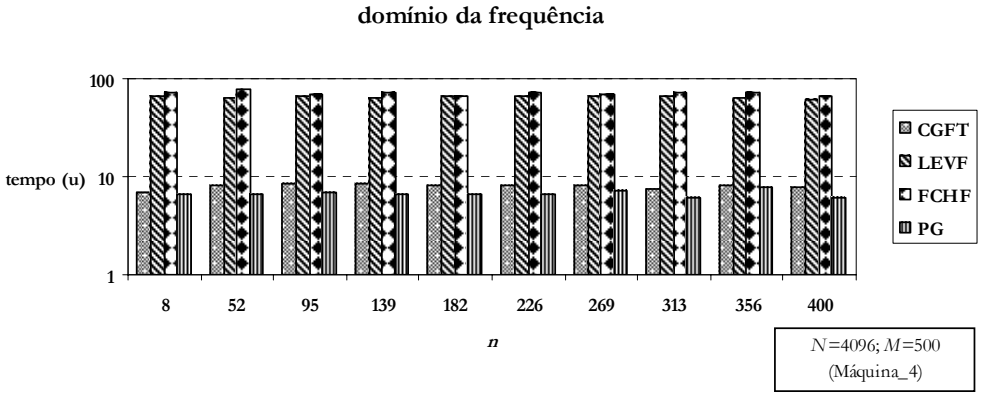

Figura 6.13: Métodos na frequência e PG para Problema_1 quando o padrão de erros é aleatório.

6.3.2 Melhores desempenhos no domínio do tempo (Problema_3) em três arquitecturas

Utilizando os resultados obtidos pelo Problema_3 podemos verificar claramente que os métodos no tempo têm desempenhos melhores do que os na frequência. Neste caso o problema é bem condicionado o que implica que as matrizes dos métodos tenham números de condição pequenos. Nas figuras (6.14-19) seguintes apresentam-se resultados que comprovam estes factos em três máquinas: Máquina_4, Máquina_6 e Máquina_3.

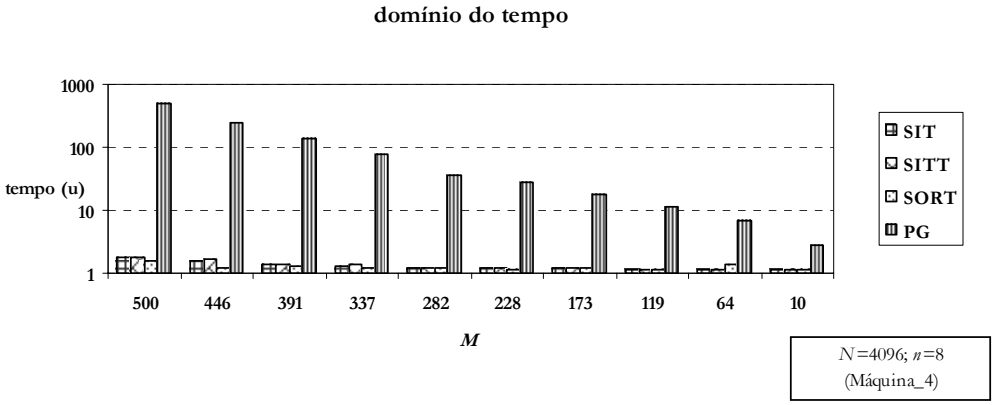

Figura 6.14: Métodos no tempo iterativos e PG para Problema_3 e Máquina_4: padrão de erros contíguo.

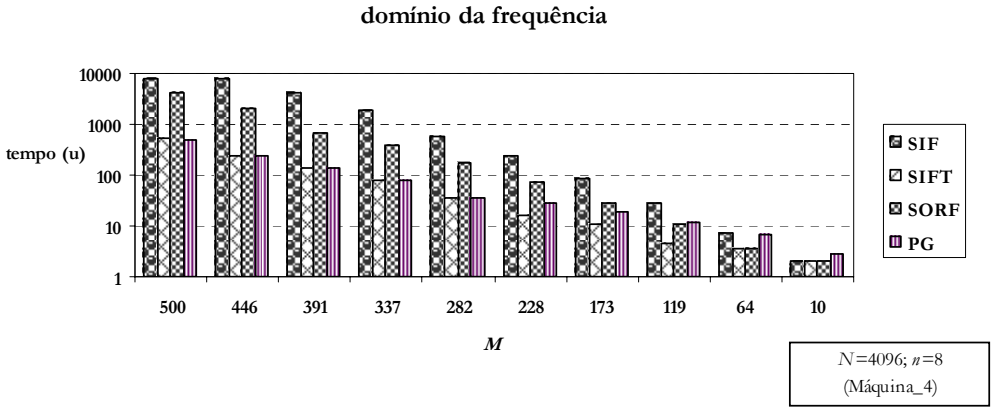

Figura 6.15: Métodos na frequência e PG para Problema_3 e Máquina_4: padrão de erros contíguo.

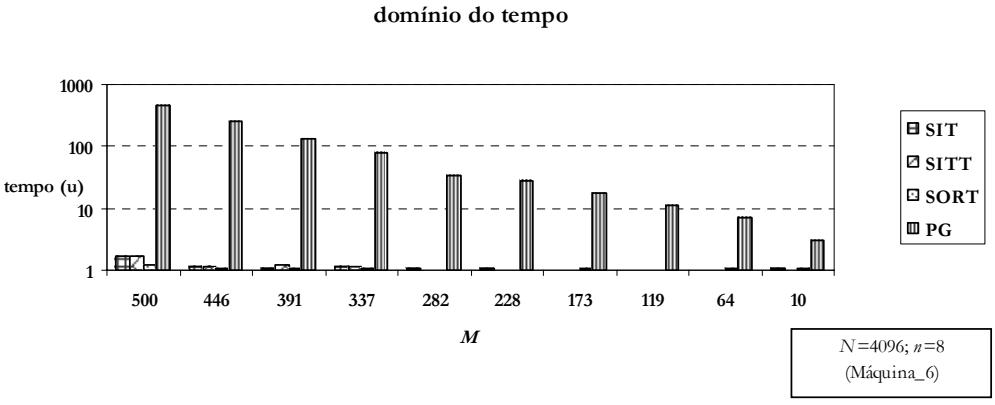

Figura 6.16: Métodos no tempo iterativos e PG para Problema_3 e Máquina_6: padrão de erros contíguo.

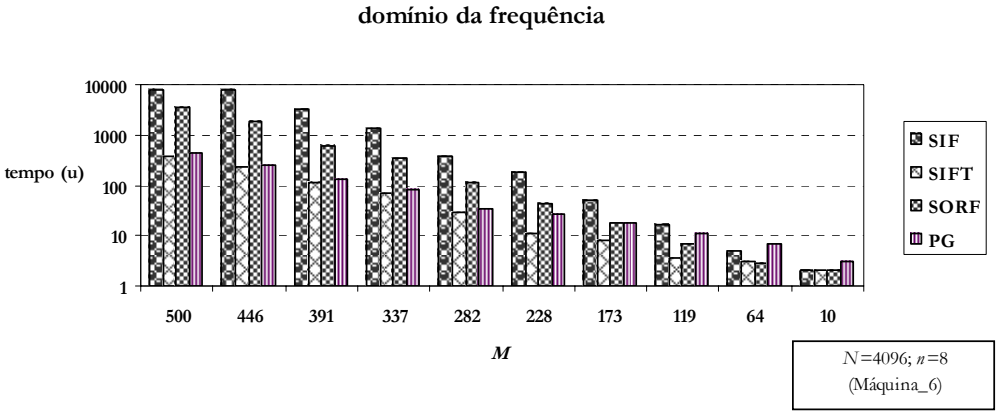

Figura 6.17: Métodos na frequência e PG para Problema_3 e Máquina_6: padrão de erros contíguo.

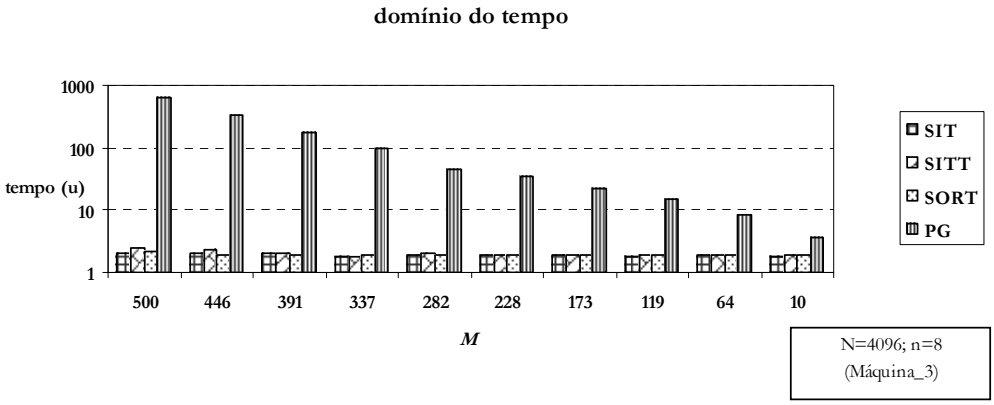

Figura 6.18: Métodos no tempo iterativos e PG para Problema_3 e Máquina_3: padrão de erros contíguo.

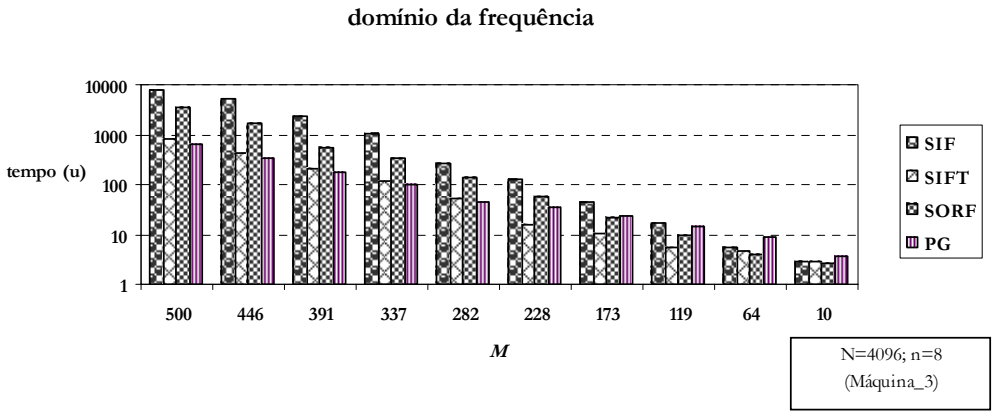

Figura 6.19: Métodos na frequência e PG para Problema_3 e Máquina_3: padrão de erros contíguo.

6.3.3 Melhores desempenhos no domínio frequência (Problema_4)

Considerando agora fixo o número de amostras perdidas, n=400, e considerando novamente um padrão de erros aleatório, se fizermos variar o grau de redundância do código no intervalo [0.25, 0.0005] podemos observar nas figuras 6.20 e 6.21 que os métodos iterativos da frequência consomem progressivamente menos CPU à medida que o código se torna mais redundante $r \approx 0.0005$ (M=10).

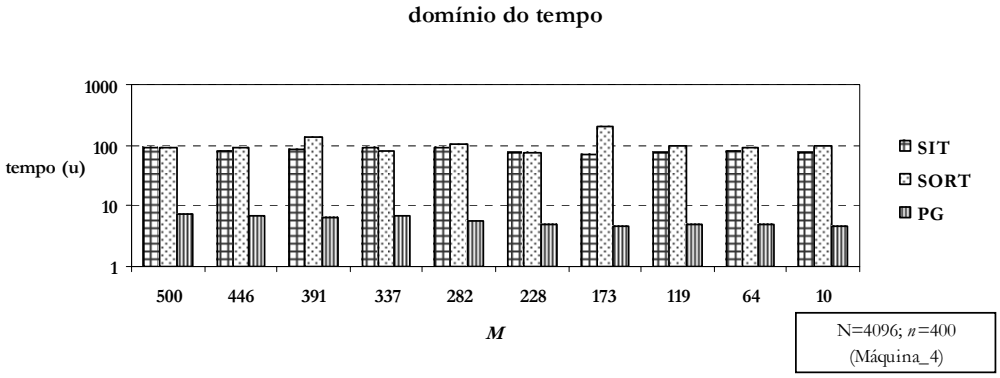

Figura 6.20: Métodos iterativos no tempo para Problema_4 quando o padrão de erros é aleatório.

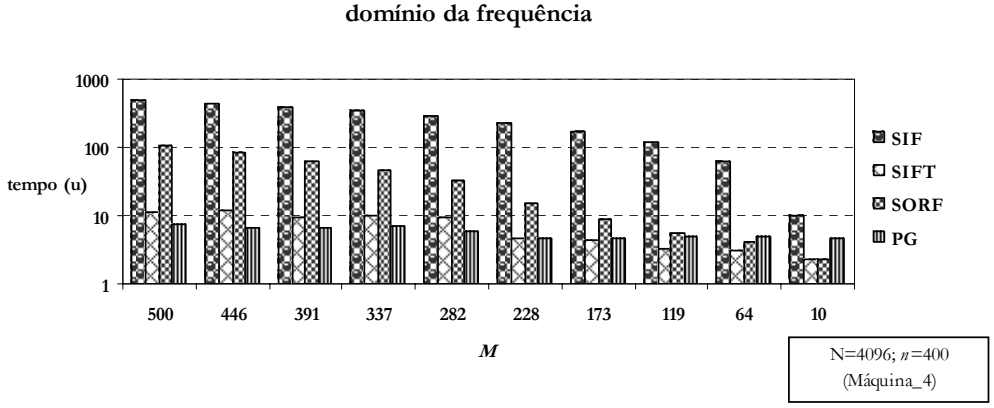

Figura 6.21: Métodos iterativos na frequência para Problema_4 quando o padrão de erros é aleatório.

Como já foi referido, este problema pretende representar uma situação limite em que agora os métodos que melhores performances apresentam são escolhidos entre os da

frequência. Este facto deve-se à ordem dos sistemas que no tempo é sempre igual ao número de amostras perdidas 400, enquanto que na frequência vai diminuindo de ($2M$+1=1001) para ($2M$+1=21), à medida que a redundância do código aumenta.

Claramente se pode concluir por análise das figuras 6.22 e 6.23 que o método óptimo para um problema com estas características teria de ser eleito a partir da formulação no domínio da frequência, por exemplo o FCHF.

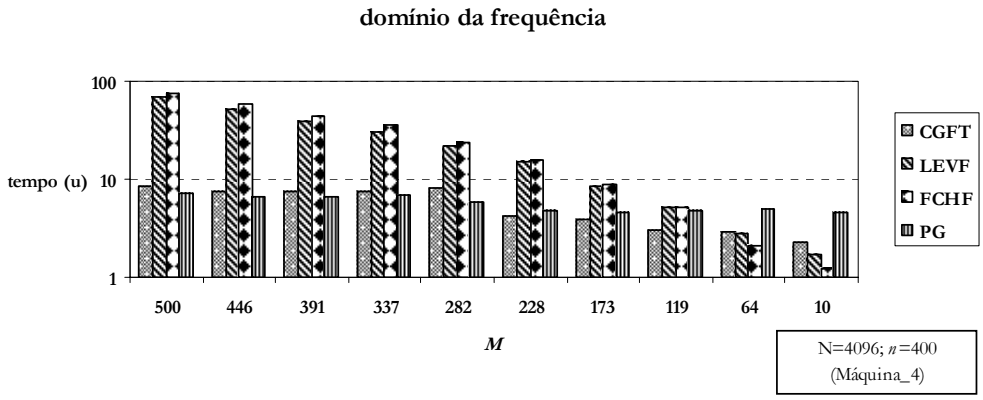

Figura 6.22: Métodos na frequência para Problema_4 quando o padrão de erros é aleatório.

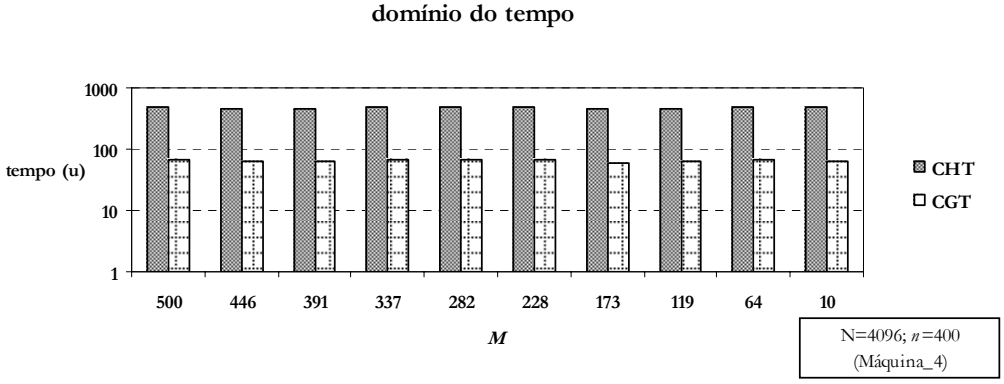

Figura 6.23: Métodos no tempo para Problema_4 quando o padrão de erros é aleatório.

6.3.4 Os pontos críticos (Problema_2)

Vamos considerar agora um código muito redundante $r \approx 0.0005$ (M=10). Fazendo variar a quantidade de amostras perdidas contíguas no tempo [8, 400], verificamos que à medida que este número aumenta, os métodos do tempo requerem progressivamente mais tempo tornando-se a partir de certo ponto mais lentos. Este facto relaciona-se com a ordem dos sistemas de equações no tempo que vai aumentando à medida que o número de amostras perdidas aumenta, figuras 6.24 e 6.25.

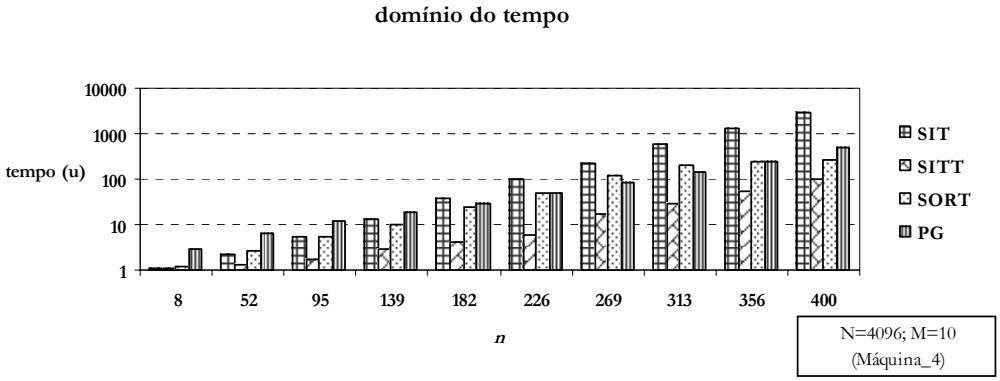

Figura 6.24: Métodos iterativos no tempo para perdas contíguas no Problema_2.

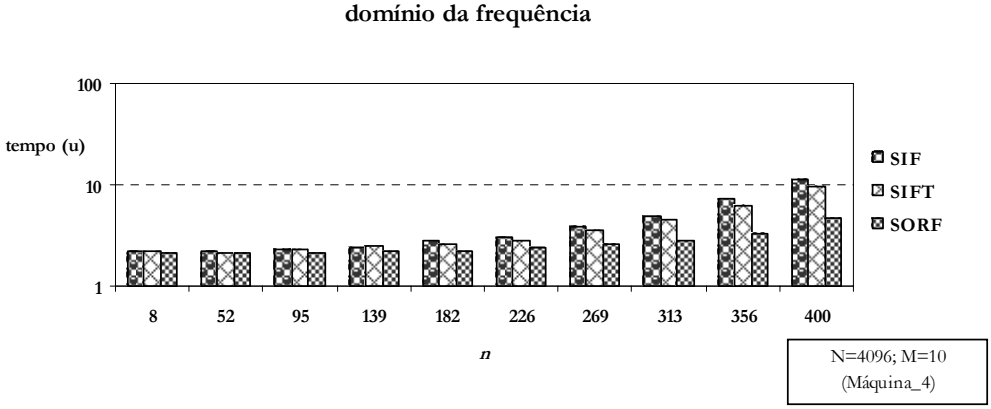

Figura 6.25: Métodos iterativos na frequência para perdas contíguas no Problema_2.

De notar que em comparação com o Problema_1 (figuras 6.4 e 6.5) onde a única diferença reside no facto do código ser menos redundante (M=500), como aqui o problema está melhor condicionado (M=10) todos os métodos apresentam melhores performances.

Apesar dos métodos na frequência necessitarem, regra geral, de menos unidades de CPU pois têm ordem pequena ($2M$+1=21), podemos constatar nas figuras 6.26 e 6.27 que para este problema alguns métodos no tempo têm performances semelhantes aos melhores na frequência.

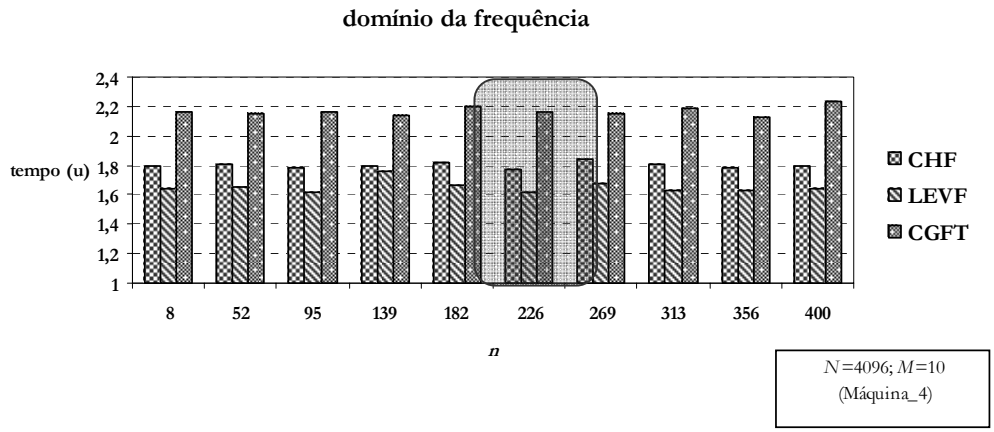

Figura 6.26: Métodos óptimos na frequência para perdas contíguas no Problema_2. A área a sombreado assinala a zona crítica.

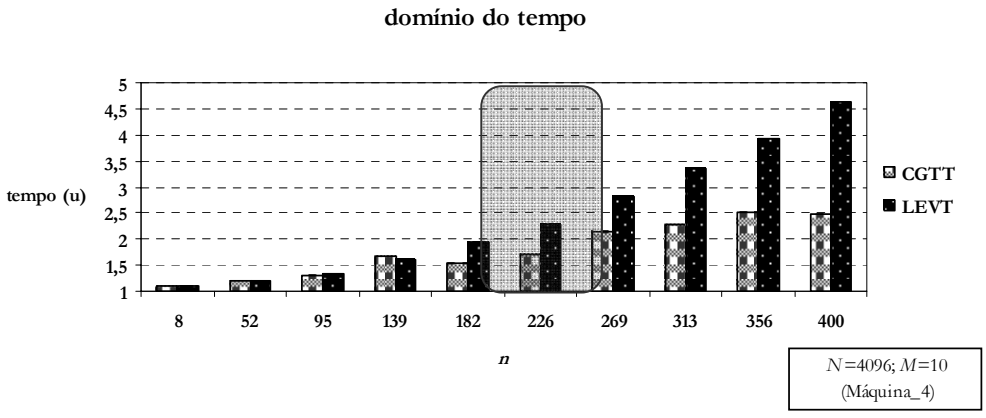

Figura 6.27: Métodos óptimos no tempo para perdas contíguas no Problema_2. A área a sombreado assinala a zona crítica.

Destacam-se o LEVT e o CGTT que mesmo com sistemas de equações de ordem 400 (mais elevada do que na frequência), têm tempos de execução comparáveis aos dos métodos na frequência em especial o CGTT. Mais uma vez se verifica que as características intrínsecas de cada problema seleccionam naturalmente uns métodos em detrimento doutros, reforçando mais uma vez a inexistência dum método ideal que resolve todos os problemas com um desempenho óptimo.

A área a sombreado nas figuras 6.26 e 6.27 permite estabelecer os limites para os quais os métodos na frequência começam a ter melhores desempenhos para este problema. Ou seja, para um número de amostras perdidas contíguas $n \approx 226$ o método LEVF deve ser a opção em relação ao que melhor desempenho obtinha no tempo o CGTT. Nesta situação o ponto crítico aconteceria para um número de amostras perdidas ≈ 226.

Claramente, como no caso anterior, a pior situação sob o ponto de vista da reconstrução acontece quando o padrão de erros é contíguo. Se agora analisarmos o Problema_2 para o caso de perdas por segmentos verificamos que todos os métodos, regra geral, precisam de menos tempo de CPU para desempenhos semelhantes aos que existem quando as perdas são contíguas, figuras 6.24 e 6.28.

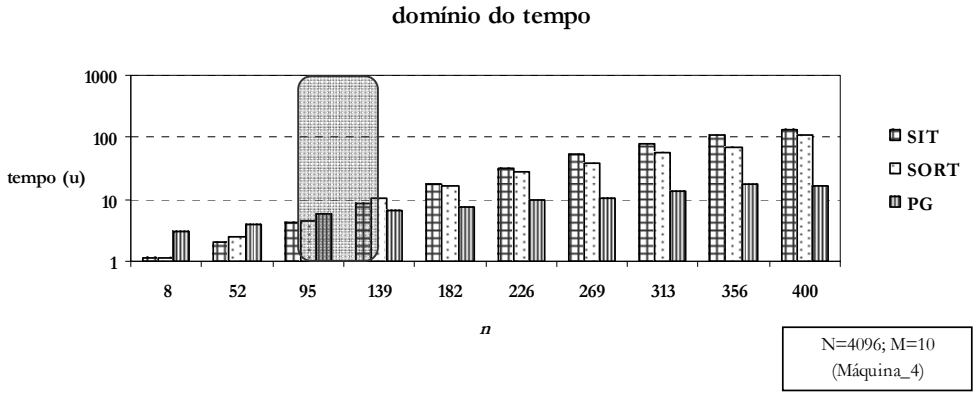

Figura 6.28: Métodos iterativos no tempo para perdas por segmentos no Problema_2. A área a sombreado assinala a zona-crítica.

De notar que os mesmos métodos têm comportamentos diferentes em função do padrão de erros nos dados, o que reforça mais uma vez o facto de não existir o método ideal que

resolve todos os problemas. Veja-se o caso do PG na figura 6.24 que começa (n=8 amostras contíguas) a consumir mais tempo de CPU do que o SORT e acaba (n=400 contíguas) também com valores mais elevados, contrariamente ao que se passa quando as perdas são por segmentos, em que claramente quando o número de amostras perdidas é igual a 400 o tempo gasto de CPU pelo PG é menor, figura 6.28.

Neste caso, o ponto crítico para o qual se deve optar entre o tempo ou a frequência corresponde a um número de amostras perdidas ≈ 95, figuras 6.29 e 6.30.

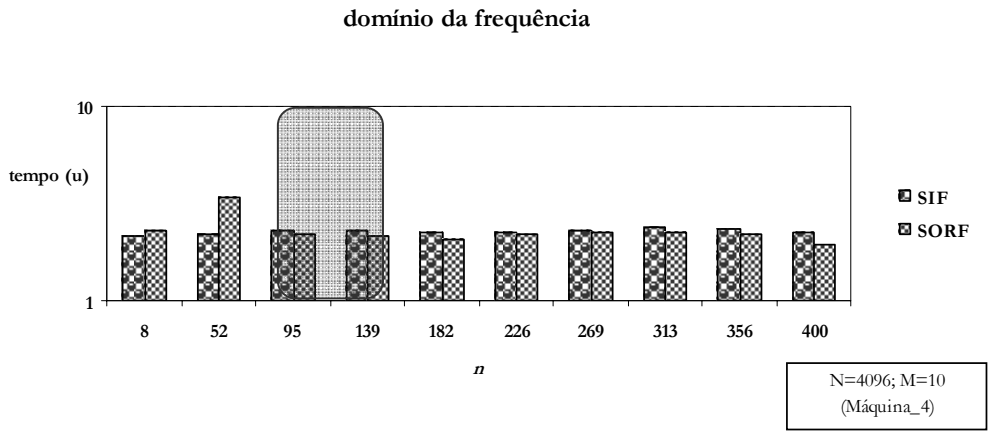

Figura 6.29: Métodos iterativos na frequência para perdas por segmentos no Problema_2. A área a sombreado assinala a zona crítica.

A melhor situação sob o ponto de vista da reconstrução para qualquer problema ocorre quando as perdas são aleatórias já que é nesta situação que existe uma dependência mais forte entre as amostras. Na figura 6.30 apresentam-se os resultados relativos ao Problema_2 quando o padrão de erros é aleatório. Pode confirmar-se que o ponto crítico, como no caso em que as amostras perdidas ocorrem por segmentos, se mantém ≈ 95 já que a partir deste valor o desempenho dos métodos baseados em gradientes conjugados na frequência começam a apresentar melhores performances.

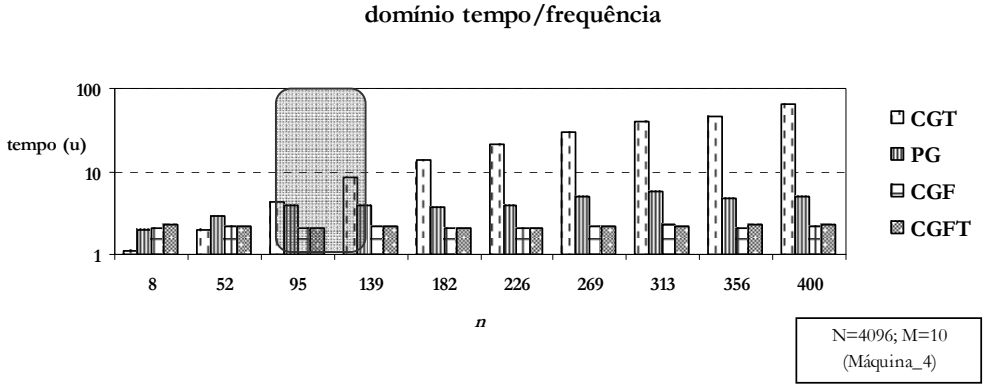

Figura 6.30: Métodos no tempo e na frequência para perdas aleatórias no Problema_2. A área a sombreado assinala a zona crítica.

6.4 Comparação com outras arquitecturas

Tomámos como referência o Problema_2 com perdas contíguas e escolhemos três métodos (dois na frequência LEVF e CGFT e um no tempo SITT) no sentido de tentar verificar se o ponto crítico se mantém aproximadamente constante ao longo das diversas máquinas.

Concluímos que o ponto crítico que permite seleccionar os métodos na frequência LEVF ou CGFT em detrimento do SITT, se mantêm em $n \approx 139$ ao longo das diversas máquinas (figuras 6.31 a 6.34).

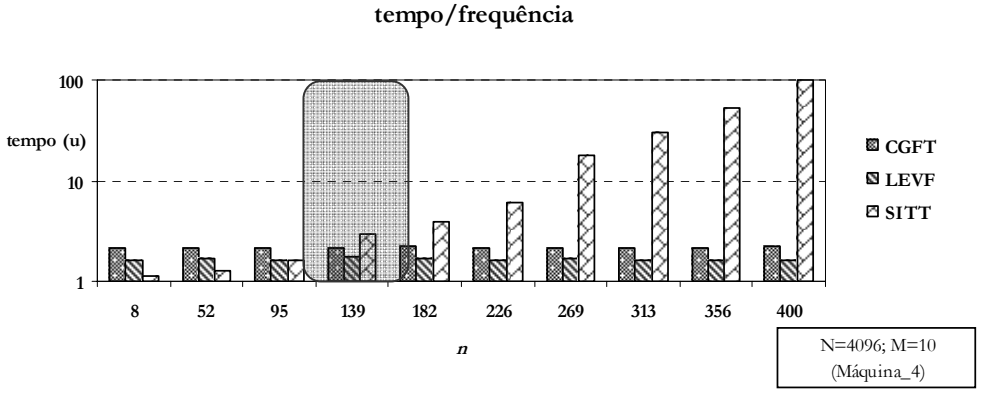

Figura 6.31: Ponto crítico para Problema_2, Máquina_4. A área a sombreado assinala a zona crítica.

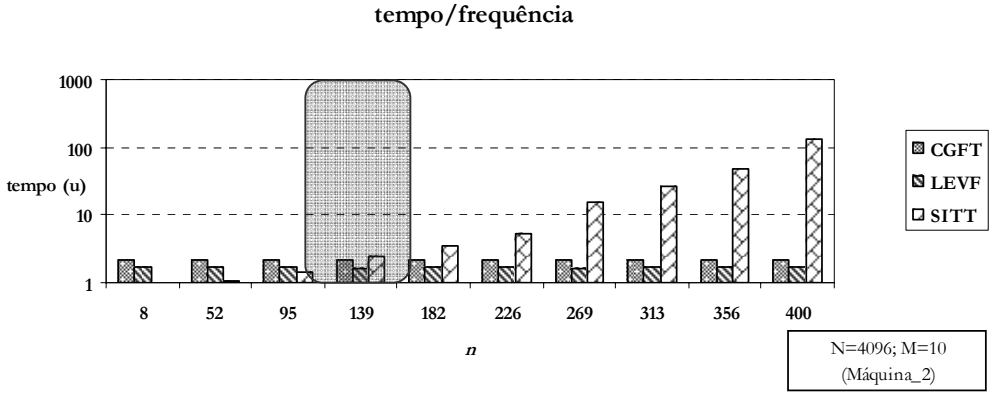

Figura 6.32: Ponto crítico para problema_2, Máquina_2. A área a sombreado assinala a zona crítica.

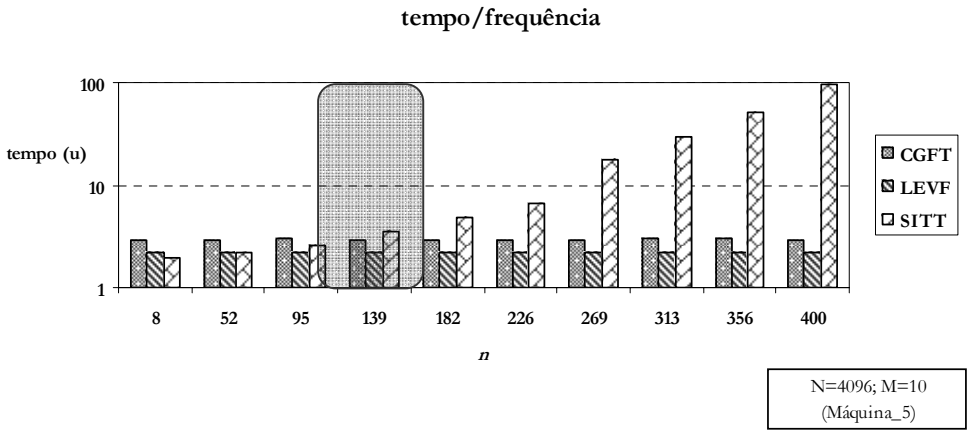

Figura 6.33: Ponto crítico para problema_2, Máquina_5. A área a sombreado assinala a zona crítica.

De notar que o facto de utilizar como unidade **1u** permite à partida verificar que para uma mesma experiência todos os métodos gastam mais ou menos a mesma quantidade de tempo máquina. Verifique-se por exemplo que o SITT consome ≈ 100**u** em todas as máquinas quando o número de amostras perdidas é igual a 400.

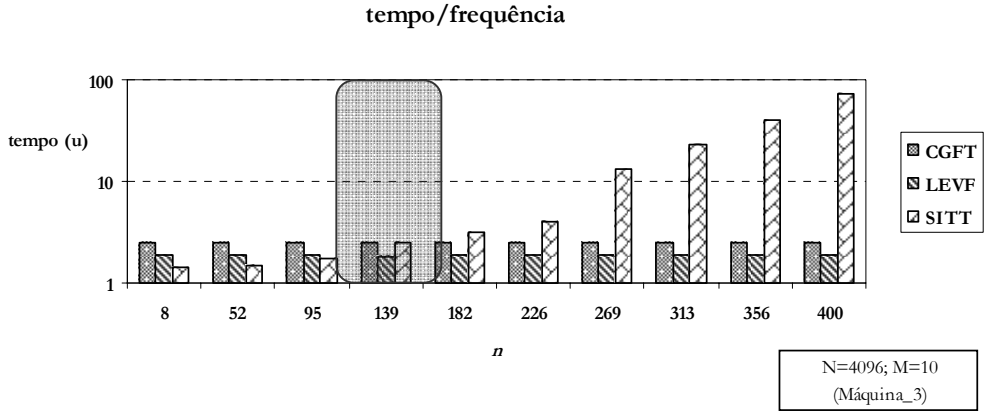

Figura 6.34: Ponto crítico para problema_2, Máquina_3. A área a sombreado assinala a zona crítica.

O facto da Máquina_6 ser a mais rápida faz com que para um número de amostras perdidas pequeno o método SITT convirja num tempo inferior a **1u**. Pode verificar-se no entanto na figura 6.35 que o ponto crítico se mantém mesmo quando comparamos o computador com o melhor e um dos piores desempenhos, Máquinas_6 e Máquina_2 respectivamente.

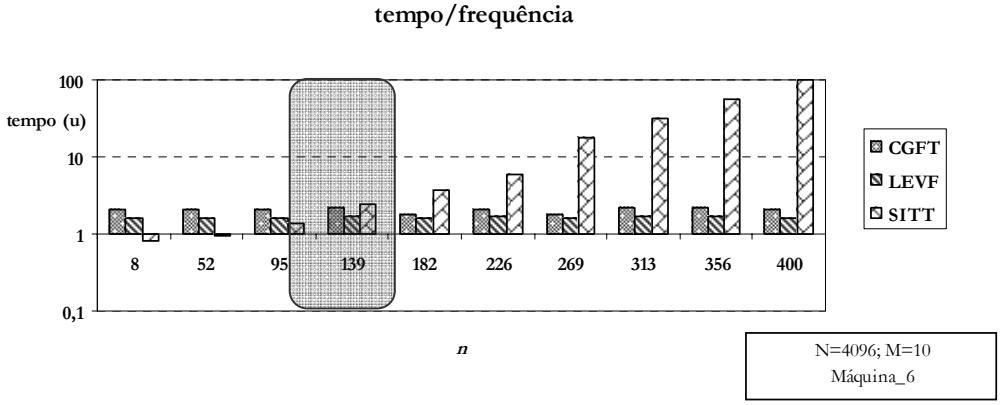

Figura 6.35: Ponto crítico para problema_2, Máquina_6. A área a sombreado assinala a zona crítica.

Foram realizados alguns testes com uma versão do compilador diferente, gcc 2.95.4, com o objectivo de verificar a consistência de algumas conclusões. Para esta experiência e para a Máquina_6 os resultados mantêm-se, mesmo para diferentes capacidades de memória cache de 512 KiB e 1024 KiB, figuras 6.36 e 6.37 respectivamente.

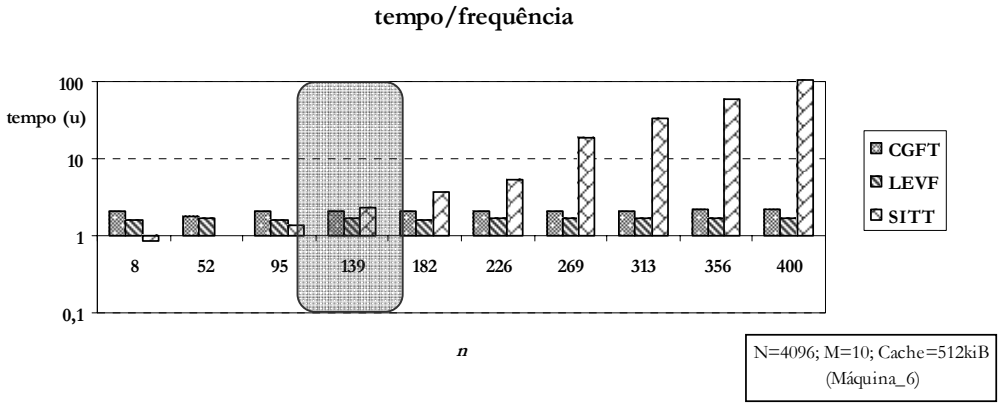

Figura 6.36: Ponto crítico para Problema_2, Máquina_6 com 512KiB de cache. A área a sombreado assinala a zona crítica.

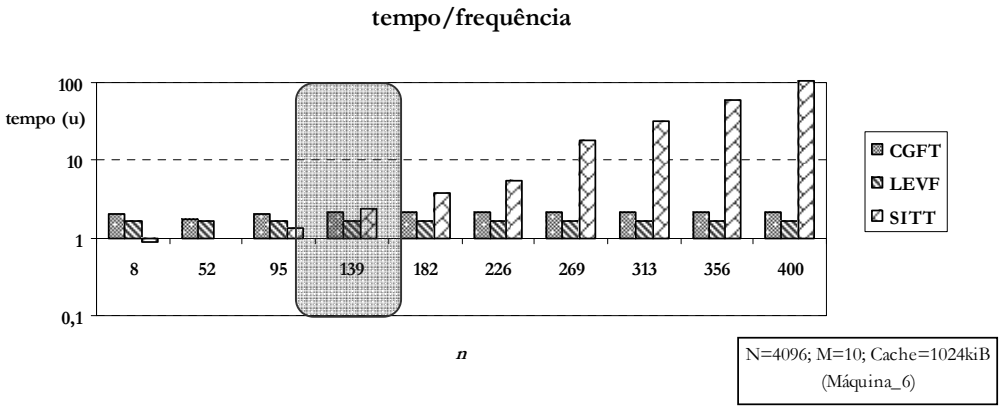

Figura 6.37: Ponto crítico para Problema_2, Máquina_6 com 1024KiB de cache. A área a sombreado assinala a zona crítica.

CAPÍTULO 7

CONCLUSÕES FINAIS E TRABALHO FUTURO

Neste capítulo, resumem-se os principais resultados deste trabalho realçando os pontos que nos parecem mais importantes ou originais.

Ulteriormente, mencionam-se alguns problemas em aberto e apresentamos algumas propostas para trabalho futuro.

7.1 CONCLUSÕES

Vimos que os algoritmos de reconstrução de sinal (interpolação ou extrapolação) podem ser interpretados como métodos de descodificação de códigos reais, e que o problema formulado neste contexto depende da resolução de um certo conjunto de equações lineares. Estas equações podem obter-se a partir de duas formulações de dimensão mínima, assim chamadas porque a ordem dos sistemas coincide com a dimensão do sub-espaço relacionado com as amostras desconhecidas, ou seja: o número de amostras desconhecidas no tempo ou o número de harmónicos não nulos na frequência.

O estabelecimento das duas formulações e a resolução das equações que delas resultam completando o conjunto de métodos com todas as formulações possíveis no tempo e na frequência constitui um dos resultados mais importantes deste trabalho. Foram estudados neste contexto os métodos na frequência Cholesky (CHF), Levinson (LEVF), Iteração Simples (SIF), e no tempo Gradientes Conjugados com aceleração Toeplitz (CGT), Gradientes Conjugados com pré-condicionamento e aceleração Toeplitz (PCGT), *Sucessive Overrelaxation* (SORF).

Concluímos que não existe um método ideal que resolva todos os problemas. No entanto, a natureza do padrão de erros é fundamental para a determinação do método mais rápido. Com erros esparsos, os métodos iterativos convergem em geral rapidamente, as matrizes dos métodos directos são bem condicionadas, e mesmo o PG pode ser eficiente (em certos casos, é mesmo o mais eficiente). Com erros contíguos torna-se possível explorar a estrutura Toeplitz das matrizes dos métodos no domínio do tempo, mas dependendo do problema os métodos iterativos podem convergir lentamente e os métodos directos podem também colocar dificuldades devido ao mau condicionamento. Os métodos de factorização podem ser altamente vantajosos caso seja necessário resolver múltiplas vezes equações com a mesma matriz (de notar que a matriz pode ou não depender do padrão de erros, dependendo da formulação escolhida).

A avaliação dos quinze métodos (tabela 6.1) de descodificação nas seis arquitecturas distintas (tabela 6.2) permitiu responder a grande parte das questões apresentadas em 6.2 e ajudou a clarificar a localização dos pontos críticos, a partir dos quais se opta por classes de métodos no tempo em detrimento dos na frequência ou vice-versa. O conhecimento acerca dos pontos críticos é fundamental porque permite manter um nível de desempenho óptimo, seleccionando o método óptimo em função das características do problema.

Apesar disto, regra geral se $2M+1 << n$ deve optar-se por métodos na frequência em detrimento dos no tempo. Caso contrário, ou seja, se $n >> 2M+1$ os métodos no tempo devem ser os eleitos.

Apesar de não vislumbrarmos uma relação directa entre as diferentes arquitecturas e a localização dos pontos críticos (secção 6.4), pensamos que a inclusão de outras arquitecturas para além da Intel (Pentium) poderia revelar novos factos.

7.2 TRABALHO FUTURO

CONCATENAÇÃO DE CÓDIGOS REAIS

Originalmente, os primeiros turbo-códigos eram constituídos pela concatenação paralela de dois códigos convolucionais simples com entrelaçamento. O conceito aqui usado, pode ser usado no contexto dos corpos reais ou complexos com o objectivo não de

aproximar o limite de Shannon, mas de melhorar as propriedades numéricas dos códigos reais como em [Ferreira03]. Uma das tentativas de compensar a instabilidade numérica dos códigos reais com um canal nomeadamente a erros do tipo rajada, é então utilizar a concatenação paralela de códigos reais com um permutador. O desempenho e estabilidade destes sistemas quando comparados com sistemas de um canal, mostram que os números de condição dos operadores lineares envolvidos podem ser menores várias ordens de grandeza em relação aos operadores nos sistemas equivalentes, mas de apenas um canal. Exemplos parecem confirmar que estes novos sistemas são mais estáveis quando sujeitos a erros do tipo rajada.

INFORMAÇÃO MULTIMÉDIA E CÓDIGOS REAIS

O começo do novo século apresenta-se-nos tecnologicamente próspero se atendermos à quantidade de aplicações que nos é oferecida nas mais variadas áreas. Algumas em fase de desenvolvimento, enquanto outras que se encontram num estádio de evolução muito adiantado, revelam o seu carácter nobre em termos sociais permitindo, por exemplo, o Ensino Electrónico à distância, a Telemedicina, ou mesmo o apoio a cidadãos com necessidades especiais. Os sistemas de navegação automática que recorrem a sistemas de informação geográfica, o comércio electrónico, as comunicações digitais onde estão incluídas as móveis, as livrarias e publicações electrónicas, a realidade virtual ou a investigação cooperativa por grupos, contam-se entre outras áreas de aplicação também importantes.

O tráfego de áudio é caracterizado por ser sensível a atrasos e tolerar taxas de erro significativas. O vídeo tem características semelhantes às do áudio no que respeita a requisitos de tempo real mas necessita de maior largura de banda, sendo a taxa de erro aceitável para o vídeo tipicamente superior à do áudio. Por seu lado, o tráfego de dados é menos sensível a atrasos, mas não tolera erros. Ao contrário dos algoritmos de compressão [Salomon00], os protocolos de armazenamento e transmissão de dados na forma digital não são na sua grande maioria sensíveis às características do tipo de dados transmitidos ou armazenados, tratando todo e qualquer tipo de dado como se fosse uma sequência de bits genérica. Este mecanismo pode tornar complexa a integração de diferentes tipos de dados. Este problema resulta essencialmente da diferente susceptibilidade aos atrasos de transmissão, à ocorrência de erros e à perda de informação. Neste contexto, e tendo atenção ao tipo de dados envolvido, os novos

campos de aplicações destes algoritmos podem estender-se ao armazenamento transmissão de informação multimédia [Akansu99, Ferreira99]. Por exemplo, nas comunicações móveis, nomeadamente quando se transmite voz e dados GPRS [Anderson01, Bates02, Soares03], pode ser possível integrar algoritmos de controlo de erros com códigos reais já que este protocolo é semelhante ao IP no que diz respeito à comutação de pacotes. Claro está que os requisitos que cada tipo de dados impõe e cada aplicação em particular deve ajustar-se no sentido da possibilidade prática de aplicação dos algoritmos. Nomeadamente em comunicações móveis a qualidade de serviço (QoS) pode ser melhorada com os códigos reais [Ferreira04].

ABORDAGEM DOS CÓDIGOS REAIS COM NÚCLEOS INTERPOLADORES DE SUPORTE COMPACTO

A avaliação da estabilidade de representações de sinais baseadas em bases não ortogonais de funções que constituam simultaneamente representações robustas, ou seja, tolerantes a erros e a ruído, pode relacionar-se com a reconstrução de sinal. A aproximação de sinais que ocorrem na prática de suporte compacto (finitos) pelas tradicionais *sinc* origina erros de representação, já que a função *sinc* tem um suporte infinito. Esta característica numa situação prática introduz sempre um erro de truncatura. Por outro lado, quando se pretende interpolar sinais sujeitos a ruído, o valor das amostras, correcto ou incorrecto, influencia o cálculo de todas as outras. Quando se usam núcleos interpoladores baseados em *splines*, sendo estes de suporte compacto, não só o erro de truncatura fica confinado aos extremos do sinal, como o valor das amostras só influencia o cálculo na região de suporte do *spline* [Ries91]. Em [Reis00] estuda-se a utilização de núcleos interpoladores de suporte compacto (como todos os sinais práticos), como é o caso das B-*splines*, em contraste com as tradicionais *sinc* na análise de Fourier para o problema da correcção de apagamentos. Em geral, podemos pensar nas amostras do sinal como pesos no processo de reconstrução. Para o núcleo $sinc(t)$ com suporte infinito os valores são todos usados sempre. Para núcleos com suporte compacto, são usados apenas aqueles que pertencem ao suporte. Contrariamente ao que acontece com o núcleo $sinc(t)$ isto implica que se as amostras tiverem valor correcto, estas influenciam positivamente a reconstrução das amostras vizinhas, sendo esta influência negativa quando estes valores forem incorrectos. O enquadramento dos

códigos reais neste contexto pode ter algumas vantagens, nomeadamente em relação à estabilidade e rapidez dos algoritmos de reconstrução.

NOVA VERSÃO DA FFTW (FFTW 3.0.1)

Os autores do algoritmo fftw, anunciam para a nova versão um incremento no cálculo da DFT na ordem dos 20%. Nos métodos onde o desempenho depende fortemente da rapidez no cálculo da DFT, como são os casos do PG e SITT, poderão ser notadas diferenças de comportamento que podem originar o deslocamento de pontos críticos.

IMPLEMENTAÇÃO EM DSPS

A realização de códigos, permite avaliar através de testes práticos (*Mean Opinion Score*, MOS, por exemplo) todas as suas capacidades [Nafie94]. Como já mencionado, o facto da aritmética dos códigos reais ser a usual e de não existirem limitações quanto ao tamanho do bloco, constituem uma mais valia destes métodos em possíveis implementações em DSPs. Para além duma necessária comparação com as outras técnicas existentes, testes em aplicações distintas de tempo real (ou não), poderão clarificar algumas questões sobre desempenho relativo que se mantêm em aberto.

APÊNDICE A

ALGORITMOS

input x, A, b, M, ε, δ

$r \leftarrow b - Ax$

$v \leftarrow r$

$c \leftarrow \langle r, r \rangle$

for k= 1 **to** M **do**

if $\langle v, v \rangle^{1/2} < \delta$ **then** *exit loop*

$z \leftarrow Av$

$t \leftarrow c / \langle v, z \rangle$

$x \leftarrow x + tv$

$r \leftarrow r - tz$

$d \leftarrow \langle r, r \rangle$

if $d < \varepsilon$ **then** *exit loop*

$v \leftarrow r + (d/c)v$

$c \leftarrow d$

output k, x, r

end do.

Algoritmo 1: Resolução de $Ax=b$ pelo método dos Gradientes Conjugados

input x, A, b, M, Q, ε, δ

$r \leftarrow b - Ax$

solve $Qz=r$ for z

$v \leftarrow r$

$c \leftarrow \langle z, r \rangle$

for k= 1 **to** M **do**

if $\langle v, v \rangle^{1/2} < \delta$ **then** *exit loop*

$z \leftarrow Av$

$t \leftarrow c / \langle v, z \rangle$

$x \leftarrow x + tv$

$r \leftarrow r - tz$

solve $Qz=r$ for z

$d \leftarrow \langle z, r \rangle$

if $d < \varepsilon$ **then**

if $\langle r, r \rangle < \varepsilon$ **then** *exit loop*

end if

$v \leftarrow r + (d/c)v$

$c \leftarrow d$

output k, x, r

end do.

Algoritmo 2: Resolução de $Ax=b$ pelo método dos Gradientes Conjugados com pré-condicionador.

APÊNDICE B

ARQUITECTURAS TESTADAS

Máquina 1:

Modelo : ***Pentium* II (*Deschutes*)**
CPU : **392.021** MHz
Cache : **512** KiB
RAM : **320** MiB
gcc : **versão 2.96**

SINGLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	173.569365	172.619944
FFTW_FORWARD, in place, generic	160.186780	159.702271
FFTW_BACKWARD, out of place, generic	172.665799	169.526530
FFTW_BACKWARD, in place, generic	160.960675	160.905158
FFTW_FORWARD, out of place, specific	172.861979	172.852756
FFTW_FORWARD, in place, specific	160.130162	159.606232
FFTW_BACKWARD, out of place, specific	172.184590	172.947082
FFTW_BACKWARD, in place, specific	160.186088	160.564584
DOUBLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	148.265726	148.335930
FFTW_FORWARD, in place, generic	124.069257	124.382738
FFTW_BACKWARD, out of place, generic	145.942127	147.154884
FFTW_BACKWARD, in place, generic	127.026187	126.833066
FFTW_FORWARD, out of place, specific	147.859926	147.610443
FFTW_FORWARD, in place, specific	125.355700	125.841161
FFTW_BACKWARD, out of place, specific	149.775913	149.501474
FFTW_BACKWARD, in place, specific	127.283844	127.176733

Máquina 2:

Modelo : **AMD-K6(tm) 3D *processor***
CPU : **350.798** MHz
Cache : **64** KiB
RAM : **256** MiB
gcc : **versão 2.96**

SINGLE_PRECISION	***HACKS***	***NO-HACKS***
FFTW_FORWARD, out of place, generic	55.153272	55.143587
FFTW_FORWARD, in place, generic	52.954779	52.973485
FFTW_BACKWARD, out of place, generic	55.907941	55.928976
FFTW_BACKWARD, in place, generic	53.714799	53.555950
FFTW_FORWARD, out of place, specific	55.508908	55.479712
FFTW_FORWARD, in place, specific	53.304308	53.328319
FFTW_BACKWARD, out of place, specific	55.947910	55.900392
FFTW_BACKWARD, in place, specific	53.682643	53.711590
DOUBLE_PRECISION	***HACKS***	***NO-HACKS***
FFTW_FORWARD, out of place, generic	53.813796	53.918799
FFTW_FORWARD, in place, generic	50.907042	50.515093
FFTW_BACKWARD, out of place, generic	53.927604	54.468098
FFTW_BACKWARD, in place, generic	50.818847	51.061503
FFTW_FORWARD, out of place, specific	52.548187	53.570921
FFTW_FORWARD, in place, specific	49.246096	50.245062
FFTW_BACKWARD, out of place, specific	53.919184	54.393863
FFTW_BACKWARD, in place, specific	50.716888	50.662176

Máquina 3:

Modelo : ***Celeron* (*Mendocino*)**
CPU : **525.017** MHz
Cache : **128** KiB
RAM : **64** MiB
gcc : **versão 2.96**

SINGLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	244.104586	243.122488
FFTW_FORWARD, in place, generic	227.890520	225.420490
FFTW_BACKWARD, out of place, generic	248.532680	246.004906
FFTW_BACKWARD, in place, generic	230.747285	230.528489
FFTW_FORWARD, out of place, specific	244.293844	243.607023
FFTW_FORWARD, in place, specific	227.757009	227.562935
FFTW_BACKWARD, out of place, specific	248.308476	246.592218
FFTW_BACKWARD, in place, specific	229.933692	229.387257
DOUBLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	239.511590	244.285731
FTW_FORWARD, in place, generic	227.577462	227.784843
FFTW_BACKWARD, out of place, generic	248.068920	247.434734
FFTW_BACKWARD, in place, generic	229.855489	230.612818
FFTW_FORWARD, out of place, specific	244.505143	244.422570
FFTW_FORWARD, in place, specific	228.056736	228.102296
FFTW_BACKWARD, out of place, specific	248.261433	248.281894
FFTW_BACKWARD, in place, specific	231.273493	230.748329

Máquina 4:

Modelo : **HP *XE*$_2$ *Pentium* III**
CPU : **450** MHz
Cache : **256** KiB
RAM : **128** MiB
gcc : **versão 2.96**

SINGLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	216.232099	215.587602
FFTW_FORWARD, in place, generic	197.881624	197.949467
FFTW_BACKWARD, out of place, generic	219.474375	220.106330
FFTW_BACKWARD, in place, generic	197.756614	199.672718
FFTW_FORWARD, out of place, specific	173.658639	212.478843
FFTW_FORWARD, in place, specific	154.974335	197.155543
FFTW_BACKWARD, out of place, specific	218.125320	214.343319
FFTW_BACKWARD, in place, specific	199.708276	194.872907
DOUBLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	195.329832	198.885286
FFTW_FORWARD, in place, generic	172.316660	181.642120
FFTW_BACKWARD, out of place, generic	203.411122	203.298267
FFTW_BACKWARD, in place, generic	182.768612	185.947614
FFTW_FORWARD, out of place, specific	198.869492	198.714884
FFTW_FORWARD, in place, specific	180.877727	182.331848
FFTW_BACKWARD, out of place, specific	203.611105	203.531105
FFTW_BACKWARD, in place, specific	186.785122	186.844713

Máquina 5:

Modelo **: AMD Duron(tm) *Processor***
CPU **: 799.623** MHz
Cache **: 64** KiB
RAM **: 128** MiB
gcc **: versão 2.96**

SINGLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	476.925379	476.936245
FFTW_FORWARD, in place, generic	444.003282	444.179870
FFTW_BACKWARD, out of place, generic	490.873678	487.861415
FFTW_BACKWARD, in place, generic	453.525571	453.430734
FFTW_FORWARD, out of place, specific	480.650452	477.803645
FFTW_FORWARD, in place, specific	444.324675	444.620986
FFTW_BACKWARD, out of place, specific	491.041379	487.771493
FFTW_BACKWARD, in place, specific	453.342230	456.080857
DOUBLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	422.116769	420.830488
FFTW_FORWARD, in place, generic	401.661681	401.083503
FFTW_BACKWARD, out of place, generic	422.586649	421.865926
FFTW_BACKWARD, in place, generic	402.795029	402.108392
FFTW_FORWARD, out of place, specific	420.878434	420.332147
FFTW_FORWARD, in place, specific	400.902303	400.456302
FFTW_BACKWARD, out of place, specific	421.959357	421.630513
FFTW_BACKWARD, in place, specific	402.234196	401.703975

Máquina 6:

Modelo **: *Pentium* IV**
CPU **: 2533.3463** MHz
Cache **: 512 /1024** KiB
RAM **: 512** MiB
gcc **: versão 2.96**

SINGLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	970.059291	975.894282
FFTW_FORWARD, in place, generic	797.921574	796.757378
FFTW_BACKWARD, out of place, generic	938.906779	967.889978
FFTW_BACKWARD, in place, generic	756.504931	790.672135
FFTW_FORWARD, out of place, specific	965.605272	981.278887
FFTW_FORWARD, in place, specific	778.840206	808.240688
FFTW_BACKWARD, out of place, specific	946.637302	966.371075
FFTW_BACKWARD, in place, specific	787.320894	791.095060
DOUBLE_PRECISION	***HACKS***	***NO_HACKS***
FFTW_FORWARD, out of place, generic	1003.096043	1004.818319
FFTW_FORWARD, in place, generic	949.165348	944.777693
FFTW_BACKWARD, out of place, generic	994.384398	997.055231
FFTW_BACKWARD, in place, generic	943.221422	944.859701
FFTW_FORWARD, out of place, specific	996.744816	1007.999635
FFTW_FORWARD, in place, specific	944.993405	947.029602
FFTW_BACKWARD, out of place, specific	1003.337669	1001.898876
FFTW_BACKWARD, in place, specific	945.182952	919.889699

BIBLIOGRAFIA

SIGLAS

ANSI	*American National Standards Institute, Inc.*.
ADC	*Analog to Digital Converter.*
ATM	*Asynchronous Transfer Mode.*
BCH	Bose, Chaudhury e Hocquenghem.
CD	*Compact Disk.*
CPU	*Core or Central Processing Unit.*
DFT	*Discrete Fourier Transform.*
DSPs	*Digital Signal Processors.*
DVD	*Digital Versatile Disk.*
FFTW	*Fast Fourier Transform in the West.*
GPRS	*General Packet Radio Services.*
IEEE	*Institute of Electrical and Electronic Engineers.*
IFFT	*Inverse Fast Fourier Transform.*
IP	*Internet Protocol*
LAN	*Local Area Network.*
LB	Largura de banda.
MFLOPS	*Million Floating Point Operations Per Second.*
MOS	*Mean Opinion Score.*
QoS	*Quality of Service.*
PAN	*Personal Area Network.*
PDS	Processamento Digital de Sinal.
POCS	*Projections Onto Convex Sets*
TCCE	Teoria dos Códigos de Correcção de Erros.

[ANSI85] ANSI/IEEE. 1985. IEEE standard for binary floating-point arithmetic. ANSI/IEEE Std. 754-1985. *IEEE*. New York.

[Akansu99] Akansu. Ali N. e Michael J. Medley. 1999. *Wavelets, Subband and Block Transforms in Communications and Multimedia. Kluwer Academic Publishers*, USA.

[Anderson95] Anderson, E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov e D. Sorensen. 1995. LAPACK User's Guide – Release 2.0. *SIAM*. Philadelphia. URL:http://www.netlib.org/lapack/lug/lapack_lug.html.

[Anderson01] Anderson01, Christoffer. 2001. *GPRS 3G Wireless Applications. Wiley.*

[Azami96] Azami, S. Zahir, Pierre Duhamel e O. Rioul. 1996. Joint source-channel coding: Panorama of methods. *CNES Workshop on Data Compression*, Toulouse, França.

[Barret94] Barret, R., M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine e H. Van der Vost. 1994. Templates for the Solutions of Linear Systems: Building Blocks for Iterative Methods. *SIAM*. Philadelphia.(http://www.netlib.org/templates/Templates.html).

[Bates02] Bates, Regis J. "Bud". 2002. *GPRS General Packet Radio Services. MacGraw-Hill Telecom.*

[Berlekamp84] Berlekamp, E. R. 1984. *Algebraic Coding Theory. Aegean Park Press,* CA - USA.

[Berman94] Berman, Ted e Jeffrey Freedman. 1994. An Importance Sampling Analysis of Noninterleaved Viterbi Decoder in an RFI Environment. *IEEE Transactions on Communications,* 40(12):3232-3237.

[Benedetto96] Benedetto, S., e G. Montorsi. 1996. Unveiling turbo codes: Some results on parallel concatenated coding schemes. *IEEE Trans. Inform. Theory*, 42(2):409-428.

[Benedetto99] Benedetto, S., e E. Biglieri. 1999. *Principles of Digital Transmission with Wireless Application. Kluwer Academic*, New York.

[Benedetto00] Benedetto, J. J. e P. J. S. G. Ferreira. 2000. *Modern Sampling Theory: Mathematics and Applications. Birkhäuser*, Boston.

[Berrou93] Berrou, C., A. Glavieux e P. Thitimajshima. 1993. Near Shannon limit error-correcting coding and decoding: Turbo-codes. In *ICC'93*, Geneve, Switzerland, pp. 1064-1070.

[Berrou96] Berrou, C., e A. Glavieux. 1996. Near Optimum Error Correcting Coding and Decoding: Turbo-codes. *IEEE Trans. On Com.*, pp. 44(10):1261-1271.

[Black95] Black, Uyless. 1995. *ATM: Foundation for Broadband Networks. Prentice Hall*, USA.

[Blahut79] Blahut, R. E. 1979. Transform techniques for error control codes. *IBM J. Res. Develop.*, 23(3):229-315

[Blahut83] Blahut, R. E. 1983. *Theory and Practice of Error Control Codes. Addison-Wesley*, USA.

[Blahut85] Blahut, R. E. 1985. Algebraic fields, signal processing, and error control. *Proc. IEEE*, 73(5):874-893.

[Blahut03] Blahut, R. E. 2003. *Algebraic Codes for Data Transmission. Cambridge University Press*, UK.

[Brison97] Brison, Owen J. 1997. *Teoria de Galois*. Universidade de Lisboa, Faculdade de Ciências, Departamento de Matemática.

[Bronson91] Bronson, R. 1991. *Matrix Methods – An Introduction, 2nd.* Ed. *Academic Press Inc.,* San Diego, CA.

[Bunch76] Bunch, James R., e Donald J. Rose. 1976. *Sparse Matrix Computations. Academic Press*, New York.

[Cao99] Cao, Meng, K. R. Subramanian, e V. K. Dubey. 1999. Concatenated Coding Scheme for DSL Channel: Effect of Interleaving. *Proc. of the Fifth Baiona Workshop on Emerging Technologies in Telecommunications,* Spain.

[Chapra88] Chapra, S. C. e R. P. Canale. 1988. *Numerical Methods for Engineers, 2nd. Ed. McGraw-Hill*, New York.

[Chui92] Chui, Charles K.. 1992. *An Introduction to Wavelets. Academic Press*, UK.

[Cover91] Cover, Thomas. M. e Joy A. Thomas. 1991. *Elements of Information Theory. Wiley Series Telecommunications*, USA.

[Cadzow97] Cadzow, James A. 1997. Signal Restoration. *In* Richard C. Dorf, editor, *The Electrical Engineering Handbook*, *IEEE Press, CRC Press.*

[Daubechies92] Daubechies, I. 1992. *Ten Lectures on Wavelets. CBMS-NSF Regional Conference Series in Applied Mathematics*, *SIAM*, Philadelphia.

[Demmel89] Demmel, J. 1989. On Floating Point Errors in Cholesky. *Courant Institute*, LAPACK Working Note 14 Aug, New York.

[Divsalar95] Divsalar, D. e F. Pollara. 1995. Turbo codes for PCS applications. *Proc. of IEEE*.

[Dongarra79] Dongarra, J. J., C. Moler, J. Bunch e G. Stewart. 1979. LINPACK User's Guide. *SIAM*, Philadelphia.

[Duffin52] Duffin, R. J. e A. C. Schafer. 1952. A class of nonharmonic Fourier series. *Transactions of American Mathematical Society*, 72:341.366.

[Durbin60] Durbin, J. 1960. The fitting of time series models. *Rev. Inst. De Stat.*, 28(3):233-244.

[Feitchinger94] Feichtinger, H. G. e K. Gröchening. 1994. Theory and practice of irregular sampling, in Wavelets: Mathematics and Applications. *J.J. Benedetto and M. W. Frazier*, *Eds. CRC*, pp. 305-363, Boca Raton, FL.

[Ferreira92] Ferreira, P. J. S. G. 1992. Incomplete sampling series and the recovery of missing samples from oversampled band-limited signals. *IEEE Trans. Signal Processing,* 40(1):225-227.

[Ferreira94A] Ferreira, P. J. S. G. e A. J. Pinho. 1994. Errorless Restoration Algorithms for Band-Limited Images. *Proceedings of the first IEEE International Conference on Image Processing ICIP-94*, pp. 157-161, Austin Texas.

[Ferreira94B] Ferreira, P. J. S. G. 1994. Non Iterative and Fast Iterative Methods for Interpolation and Extrapolation. *IEEE Transactions on Signal Processing,* 42(11):3278-3282.

[Ferreira94C] Ferreira, P. J. S. G. 1994. Interpolation and the Discrete Papoulis-Gerschberg Algorithm. *IEEE Transactions on Signal Processing,* 42(10):2596-2606.

[Ferreira94D] Ferreira, P. J. S. G. 1994. The stability of a procedure for the recovery of lost samples in band-limited signals. *Signal Processing,* 40(3):195-205.

[Ferreira96] Ferreira, P. J. S. G. 1996. Interpolation in the Time and Frequency Domains. *IEEE Signal Processing Letters,* 3(6):176-178.

[Ferreira97] Ferreira, P. J. S. G. 1997. The Eigenvalues of Matrices that Occur in Certain Interpolation Problems. *IEEE Transactions on Signal Processing,* 45(8):2115-2120.

[Ferreira99] Ferreira, P. J. S. G. 1999. Mathematics for multimedia signal processing II – discrete finite frames and signal reconstruction. *Signal Processing for Multimedia,* pages 35-54. *IOS Press*, *J. S. Byrnes*, *Ed.*

[Ferreira00] Ferreira, P. J. S. G. 2000. Stability issues in error control coding in the complex field, interpolation and frame bounds. *IEEE Signal Processing,* 44:16-31.

[Ferreira03] Ferreira, P. J. S. G. e José M. N. Vieira. 2003. Stable DFT Codes and Frames. *IEEE Signal Processing Letters,* 10(2):50-53.

[Ferreira04] Ferreira, P. J. S. G., Dorabella M. S. Santos e Vera M. A. Afreixo. 2004. New Results and Open Problems in Real-Number Codes. *Proceedings* EUROSIPCO-2004*,* (*Special Session: Robust Transmission of Multimedia Content*), Vienna, Austria.

[Frigo99] Frigo, M. e S. G. Johnson. 1999. FFTW User's Manual. *Massachusetts Institute of Technology*.

[Frias97] Frias, J. Garcia e J. Villasenor. 1997. Joint Source Channel Decoding of Turbo Codes. *Proc. Of Int. Symp. On Turbo Codes*, *pp.* 259-262, Brest, França.

[Gerchberg74] Gerchberg, R. W. 1974. Super resolution through error energy reduction. *IEEE Opt. Acta*, 21(9):709-720.

[Golub89] Golub, G. H. e C. F. van Loan. 1989. *Matrix Computations*. Baltimore, MD: *Johns Hopkins University Press*.

[Gröchenig93A] Gröchenig, K. 1993. A discrete theory of irregular sampling. *Linear Algebra Appl.*, (193):29-150.

[Gröchenig93B] Gröchenig, K. 1993. Acceleration of the frame algorithm. *IEEE Trans. Signal Proc.* , 41(12):3331-3340.

[Hagenauer96] Hagenauer, J., E. Offer, e L. Papke. 1996. Iterative decoding of binary block and convolutional codes. *IEEE Trans. Inform. Theory*, 42(2):429-445.

[Hamming50] Hamming. R. V. 1950. Error detecting and error correcting codes. *Bell System Technical Journal*, (29):147-160.

[Heegard99] Heegard, Chris. e Stephen B. Wicker. 1999. *Turbo Coding. Kluwer Academic Publisher*, USA.

[Hestenes52] Hestenes, M. e E. Stiefel. 1952. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards*, 49(6):409-436.

[Horn85] Horn, R. A. e C. R. Johnson. 1985. *Matrix Analysis. Cambridge University Press*, New York.

[Horn91] Horn, R. A. e C. R. Johnson. 1991. *Topics in Matrix Analysis. Cambridge University Press.*

[Jones86] Jones, M. C. 1986. The Discrete Gerchberg Algorithm. *IEEE Transactions on Acoustics Speech and Signal Processing*, 3(3):624-626.

[Kaiser94] Kaiser, Gerald. 1994. *A Friendly Guide to Wavelets*, *Birkhäuser*.

[Kay88] Kay, S. M. 1988. *Modern Spectral Estimation*, *Theory & Application*. *Prentice Hall*, *Signal Processing Series*, *Alan V. Oppenheim, Series Editor*.

[Kincaid02] Kincaid, D. e W. Cheney. 2002. *Numerical Analysis: Mathematics of Scientific Computing. The Brooks/Cole Series in Advanced Mathematics, Paul J. Sally, Jr., Editor, SIAM*, USA.

[Kleijn95] Kleijn, W. B. e K.K. Paliwal. 1995. Speech Coding and Synthesis. *Elsevier*.

[Klir95] Klir, George J. e Bo Yuan. 1995. *Fuzzy Sets and Fuzzy Logic Theory and Applications*. *Prentice Hall*.

[Kumaresan86] Kumaresan, R. 1986. Rank reduction techniques and burst error-corrections decoding in real/complex fields. *In Proc. of the 19th Asilomar Conf. Circuits Syst.,* pages 457-461, Pacif Grove.

[Levinson47] Levinson, N. 1947. The Winner (root mean square) error criterion in filter design and prediction. *Journal Math. Phys.*, 25:261-278.

[Looney97] Looney, Carl G. 1997. Noise. *In* Richard C. Dorf, editor, *The Electrical Engineering Handbook*, *IEEE Press, CRC Press*, 1642:1653.

[Marks93] Marks II, R. J. 1993. *Advanced Topics in Shannon Sampling Interpolation Theory*, *Springer*, Germany.

[Marvasti97] Marvasti, F., M. Echhart, e M. Hasan. 1997. Burst correction for missing samples. *In Proc. Of the 1997 Workshop on Sampling Theory and Applications, SampTA-97*, pages 161-167, Aveiro, Portugal..

[Marvasti99] Marvasti, F., M. Hasan, M. Echhart, e S. Talebi. Efficient algorithms for burst error recovery using FFT and other transform kernels. *IEEE Trans. Signal. Processing* 47(4).

[Marshall82] Marshall, T. G. Jr. 1982. Codes and algorithms for simultaneous error correction and rate implementable with standard digital signal processors. *In Proc. Of the IEEE Global Telecommunications Conf.,* pages 936-940, Miami.

[Marshall84] Marshall, T. G. Jr. 1984. Coding of real-number sequences for error correction: A digital signal processing problem. *IEEE J. Select. Areas Commun.*, 2(2):381-391.

[Marshall86] Marshall, T. G. Jr. 1986. Codes for error correction based upon interpolation of real-number sequences. *In Proc. of the 19th Asilomar Conf. Circuits Syst.,* pages 202-206, Pacif Grove.

[Nafie94] Nafie, M., e F Marvasti. 1994. Implementation of recovery of speech with missing samples on a DSP chip. *Electronics Letters,* 30(1):12-13.

[Ott76] Ott, Henry W. 1976. *Noise Reduction Techniques in Electronic Systems. Bell Telephone Laboratories.*

[Papoulis75] Papoulis, A. 1975. A New Algorithm in Spectral Analysis and Band-Limited Extrapolation. *IEEE Transactions on Circuits and Systems*, 22(9):735-742.

[Pei97] Pei, S. C., e M. H. Yeh. 1997. An introduction to finite frames. *IEEE Signal Processing Mag.*, 14(6)84-96.

[Perez96] Perez, L. C., J. Seghers, e D. J. Costello, Jr. 1996. A distance spectrum interpretation of turbo codes. *IEEE Trans. Inform. Theory*, 42(6):1698-1709.

[Pina95] Pina, H. 1995. *Métodos Numéricos. McGraw-Hill*, Portugal.

[Press92] Press, W. H., S. A. Tenkolsky, W. T. Vetterling e B. P. Flanney. 1992. *Numerical Recipes in C – The Art of Scientific Computing- 2 nd. Ed.. Cambridge University Press.*

[Prony95] Prony, R. 1795. Essai experimental et analytique: Sur les lois de la dilatabilité des fluides élastiques et sur celles de la force expansive de la vapeur de l'eau et de vapeur de l'alkool, à différentes températures. *J. École Polytechnique Paris,*1:24-76.

[Proakis95] Proakis, John G. 1995. Digital Communication. *MacGraw/Hill.*

[Rabiner78] Rabiner, L. R. e R. W. Schafer. 1978. Digital Processing of Speech Signals. *Bell Laboratories Inc, Prentice-Hall.*

[Rath02] Rath, Gagan e Christine Guillemot. 2002. Performance Analysis and Recursive Syndrome Decoding of DFT codes for Bursty Erasure Recovery. *IEEE Trans. Signal Proc.*, 51(5)1335-1350.

[Reis00] Reis, M. J. C. S. 2000. *Representação multi-resolução de sinais baseados em Splines e sobre-amostragem. Tese de Doutoramento, UA,* Portugal.

[Ries91] Ries, Sigmar. 1991. On the reconstruction of signals by a finite number of samples. *Signal Processing*, 23(1):45-68.

[Rotman98] Rotman, Joseph. 1998. *Galois Theory. Springer*, New-York.

[Rothweiller99] Rothweiller, J. 1999. Turbo codes – making communications more efficient. *IEEE Potencials.*

[Rudin87] Rudin, Walter. 1987. *Real and Complex Analysis. McGraw-Hill.*

[Salomon00] Salomon, David. 2000. *Data Compression*. Springer, USA.

[Sanz83] Sanz, Jorge L. C. e Thomas S. Huang. 1983. Some Aspects of Band-Limited Signal Extrapolation: Models, Discrete Approximations, and Noise. *IEEE Transactions on Acous. Spe. And Sig. Proc.*, 31(6):1492-1501.

[Schafer81] Schafer, R. W., R. M. Mersereau e M. A. Richards. 1981. Constrained iterative restoration algorithms. *Proc. IEEE*, 69(4):432-450.

[Schlebush85] Schlebush, H. J. e W. Splettstöber. 1985. On a conjecture of J. L. C. Sanz and T. S. Huang. *IEEE Transactions on Acoustic, Speech, and Sinal Processing*, 33(4):1628-1630.

[Schwartz90] Schwartz, Mischa. 1990. Information Transmission, Modulation, and Noise. *McGraw-Hill*.

[Shannon48] Shannon, C. E. 1948. A mathematical theory of communication. *Bell System Technical Journal*, (27):379-423.

[Shannon49] Shannon, C. E. 1949. Communication in the presence of noise. *Proc. Of the IRE*, (37):10-21.

[Shewchuk94] Shewchuk, J. R. 1994. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. *School of Computer Science*, Carnegie University.

[Sluis92] Sluis, A. Van der. 1992. The Convergence Behaviour of Conjugate Gradients and Ritz Values in Various circumstances. *Iterative Methods in Linear Algebra*, pp. 49-66.

[Soares98] Soares, S. F. S. P. e P. J. S. G. Ferreira. 1998. Interleaving and Interpolation for the Recovery of Speech Packets on a Ethernet-Based Audio Transmission System. *WAC'98*, Alaska.

[Soares03] Soares, S, Carlos Serôdio, António Coelho, Pedro Mestre e Pedro Pinto. 2003. *Audio Interface for Real Time Navigation. 4th Conference on Telecommunication*, Aveiro.

[Stallings88] Stallings, William. 1988. Data and Computer Communication. *Mcmillan Publishing Company*.

[Steen94] Steen, A. J. V. D. 1994. Overview of recent supercomputers. Publication of the NCF, *Academic Computing Centre Utrecht*, Netherlands.

[Strohmer93] Strohmer, T. 1993. *Efficient methods for digital signal and image reconstruction from nonuniform samples. Ph.D. Institut für Mathematik der Universität Wien*, Vienna, Austria.

[Strohmer95] Strohmer, T. 1995. On discrete band-limited signal extrapolation. *in Proc. Conf. Math. Anal. Signal Processing*, Cairo, Egypt, *AMS Contemporary Mathematics*, 190:323-337, C. Chui, Ed.

[Tarczynsky97] Tarczynsky, A. 1997. Sensitivity of signal reconstruction. *IEEE Signal Processing Letters*, 4:192-194.

[Vieira97] Vieira, J. M. N. e P. J. S. G. Ferreira. 1997. Interpolation, spectrum analysis, error-control coding, and fault-tolerant computing. *in Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing, ICASSP 97*, (III):1831-1834.

[Vieira00] Vieira, J. M. N. 2000. *Reconstrução de sinal e codificação. Tese de doutoramento*, *UA*, Portugal.

[Vucetic00] Vucetic, Branka e Jinhong Yuan. 2000. *Turbo Codes Principles and Applications. Kluwer Academic Publishers*, USA.

[Youla82] Youla, Dante C. e H. Webb. 1982. Image Restoration by the Method of Convex Projections: Part I Theory. *IEEE Trans. On Medical Imaging*, 2(1):81-94.

[Young71] Young, D. M. 1971. *Iterative Solution of Large Linear Systems. Academic Press.* Orlando.

[Yu96] Yu, Gongsan, Michael W. Marcellin e Max M.-K. Liu. 1996. Recovery of video in the presence of packet loss using interleaving and spatial redundancy. *IEEE Transactions on Communications*, 40(12):3232-323.

[Wang99] Wang, G. e W. Ham. 1999. Minimum error bound of signal reconstruction. *IEEE Signal Processing Letters*, 6(12):309-311.

[Wolf83] Wolf, J. K. 1983. Redundancy, the discrete Fourier transform, and impulsive noise cancellation. *IEEE Trans. Commun.*, 31(3):458-461.

[Zhang92] Zhang, H. M. e P. Duhamel. 1992. On the method for solving Yule-Walker equations. *IEEE Transactions on Signal Processing*, 40(12):2937-3000.

[Zayed93] Zayed, A. I. FL. 1993. *Advances in Shannon's Sampling Theory. CRC Press Inc.*